Planning Discourse by Modelling Audience Interpretation Strategies

Richard James Caley

Thesis submitted for the degree of Ph.D.
University of Edinburgh
1991

Abstract

One of the major problems facing designers of computer systems which are to use natural language to achieve their aims is that of finding a way to integrate all of the information which is needed to produce reasonable texts. Amongst these types of information, one of the most complex and hardest to use is knowledge of the intended audience of the text. This thesis proposes an architecture for text planning which places the text planning system's model of its intended audience at the heart of the planning system.

The proposed architecture is based upon ideas from research into planning in Artificial Intelligence which are extended to cover some simple types of multi-agent planning. This planning model and the representation which supports it are first discussed informally and then presented in a fairly formal manner. Various simple types of audience model are then encoded within the model and discussed in terms of their effects on the linguistic competence of the resulting text planner.

A computer system named Riple has been implemented which embodies the mechanisms discussed in this thesis.

For Ishten of Mount Ming,
who reminded me to post the application form,
and who put up with me for so long.

Acknowledgements

First and foremost I wish to gratefully acknowledge the help and support of Dr Graeme Ritchie who has not only performed all the usual tasks of a PhD supervisor, but has also been willing to perform proof reading far beyond the call of duty.

My second supervisor David Moffat has also been a great help in the production of this thesis and has also been a friend when I needed to think of other things.

Of course not even the best supervisors could totally counter my ability to produce text almost, but not quite, totally unlike English. Any remaining errors are therefore all my own work.

Thanks Mum for everything from the essential ontological precondition through dragons, household tips and Garfield to patience with lack of letters and never knowing what to say when people asked what I was actually doing.

Angela Gilham has provided long letters, pleasant company, viscous comments, book and music recommendations and a connection to the real world beyond AI. Slainté.

Gary Roberts has done much towards keeping me sane over the last five years by being willing to talk about many thing unconnected to either his work or mine when I needed distracting and by putting up with the remains of innumerable culinary experiments found lurking in the kitchen.

Collin Williams, Mike Cameron-Jones, Paul McIlvenny, Andrew Robertson and others who braved the PhD 1/2/3/4/5 lunches helped immensely by letting me know that I was not alone.

Many thanks to Judy Delin, Robert Dale, Geraint Wiggins and Ewan Klein for sitting through non too coherent talks and so helping me find out what I was talking about.

Numerous people in the Department of Artificial Intelligence have provided me with ideas and criticisms. The less concrete contribution of those whose work has no direct relevance to mine, but whose presence goes to make the department such

an interesting place must also be acknowledged. I hope I have provided some help to someone in return.

The Centre for Speech Technology Research provided me with both the means to keep body and soul together and a pleasant working environment. Special thanks to the people I have worked with on the text-to-speech project — Nick Campbell, Andie Chima, Rick Innis, Steve Isard, Alex Monaghan, John Rye, Paul Taylor, Jo Verhoeven and David Walker — and to Professor Mervyn Jack for his understanding in allowing me to use CSTR resources to work on this thesis.

The Science and Engineering Research Council provided support for this work in the form of studentship no. 8531207x. Early drafts of this thesis were written using equipment donated by the Rank Xerox Ltd. University Grants Programme.

Finally, I would like to thank the many too often unsung heroines and heroes whose never ending task it is to keep the world turning. I would especially like to mention (in no particular order) Jean Bunten, Millie Tupman, Roger Burroughes, Olga Franks, Andie Ness, Tim Bradshaw, Bob Anstruther, Alison Gardiner and Janice Ovenstone whose efforts have most directly helped me many times. Many apologies to anyone I may have forgotten, all too fallible memory is one of the reasons your work has been so important.

Declaration

This thesis was composed by me and describes original work which was executed by me.

Table of Contents

List of Figures

Chapter 1

Introduction

§1 The Problem

This thesis addresses the problem of how an agent in an environment containing other agents can choose a sequence of linguistic actions which when performed will achieve some goal. In such a situation the agent must be able to take into account and manipulate the behaviour of the other agents.

It is a commonplace in the literature on natural language generation in artificial intelligence that in order to produce fluent, readable text a system must have knowledge about the beliefs, expectations and abilities of its intended audience; however little is known about how this knowledge should be used. Often a speaker's knowledge of their audience is assumed to act as a filter, taking the text which the speaker plans and modifying it to make it more acceptable to the audience. Figure 1–1 shows the structure of this kind of system. In such a system

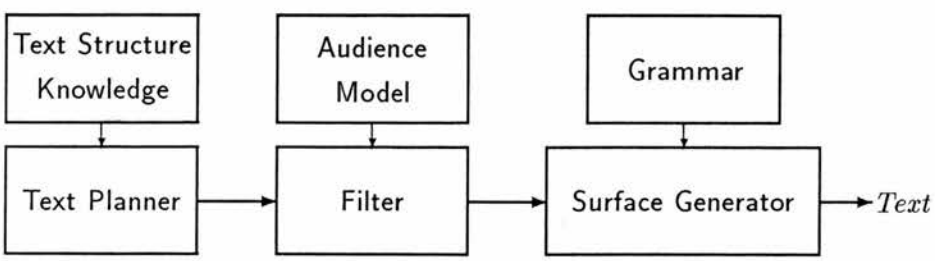


Figure 1–1: Audience Knowledge as a Filter

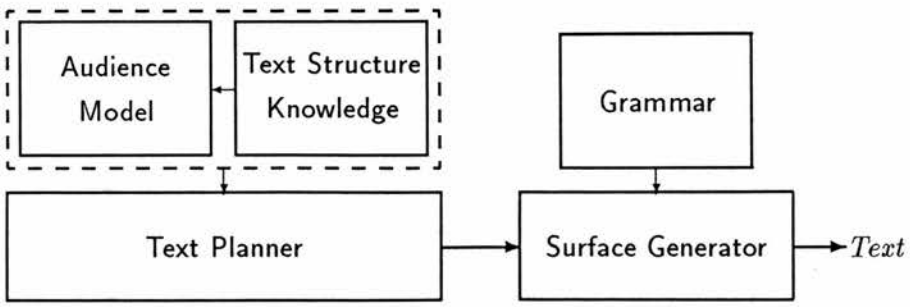


Figure 1–2: Audience Knowledge as Source of Suggestions

the audience model provides an extra set of constraints on the space of possible texts which will satisfy the speaker's goals.

In this work we present an alternative model of language generation where the audience model is seen not as a source of constraints limiting the choice of texts, but rather as a source of suggestions for possible text structures. The structure of the model developed here is shown in figure 1–2.

The work presented here is not concerned with the surface forms of the final text, only with its structure and content and how this can be chosen using knowledge of the intended audience. While it would be possible to extend the model presented here to make decisions about the surface form of the final text, effectively transferring the grammatical knowledge of the system into the audience model, it would seem that there is little to be gained by doing this.

In order to further limit the problem to a manageable size, we examine only fairly simple texts, concentrating on the interaction between speaker and audience[1] rather than on the subtleties of complex text structure. In particular we deal with only a few linguistic phenomena where the interaction is particularly important. These phenomena are —

Reminders That is, utterances[2] whose intent is not to give the audience

[1]From now on we shall always refer to the person or program *using* language as the "speaker" whether the language is spoken or written. Similarly the person or program whom the speaker is addressing will be referred to as the "audience". "Agent" will be used to mean "person or program"

[2]Utterance is used here to mean the lowest level of text structure. In this work this is taken to be the level of things which could be expressed in a single clause.

new information, but to bring some already known fact 'to mind'.

Text Structure and Focus Shift To make text comprehensible, the speaker must structure it so that related concepts are discussed together and there is a natural flow from one such topic area to the next.

Cancelling Inferences It is often necessary to make sure that the audience does not infer certain things which might be plausible inferences from what is said. For instance the desired interpretation of later utterances might be blocked by such an incorrect inference or the inference might result in the audience taking undesirable actions[3].

Disambiguating Utterances It is often necessary for the speaker to provide information in a text which is not directly related to the *goal of the text, but rather is present to provide a context* in which other parts of the text can be unambiguously interpreted.

All these phenomena involve attempts by the speaker to control the interpretation process of the audience. Such control is necessary for competent language use.

With these restrictions in mind we can define the problem of text planning more fully. We assume that the inputs to the text planning process are as follows —

- A description of the state of the world at the time when the text must be produced.
- A description of the beliefs of the intended audience of the text.
- A description of the linguistic and reasoning abilities of the intended audience.

---

[3]Of course the most common reason for a human speaker to wish to block unwarranted inferences is politeness. A speaker who allowed an audience to infer something which was not true would be held to have misled the audience as surely as if they had lied directly. However this kind of social motivation is far outside the scope of this work

- A description of how the audience can be expected to behave, for instance under what circumstances will they adopt a new goal.

- The purpose for which the text is being planned. Commonly this will be to have the audience believe some state of affairs holds.

The 'result' of text planning will be a sequence of utterances described at some suitable level of abstraction. These utterances must be such that, given the description of the world and the audience presented to the system, they will result in the purpose of the text being achieved.

Alongside its practical aim of developing a way of integrating an audience model into the text planning process, the work presented here attempts to explore the relationship between linguistic behaviour and more general rational action. While it seems certain that human linguistic behaviour is the product of specialised abilities, it would be extremely surprising if that behaviour was not also shaped by more general constraints on reasoning and rational action. In an attempt to draw out some of these connections, this work takes a single well known model of rational action, that developed in the Artificial Intelligence community under the name 'planning', and uses it as a framework upon which to base the model of text planning. The result is an exploration of how well the problems of text planning and audience modelling can be addressed by the planning model and what features of human texts can be explained as being natural consequences of general rational behaviour.

A final thread in this work is the development of a simple model of text structure and interpretation. While this model is not intended to be an interesting contribution in itself, it is capable of exhibiting interesting interpretation behaviour and it does so within the limits imposed by the needs of the text planning process.

§2 The Work

The research described here falls into a number of parts —

1. A model of text structure based on previous work by linguists but informed by the needs of the planning process.

2. A representation which can be used to represent states of the world and the beliefs of agents in the world.

3. A notation for describing domains of interest and models of agent behaviour in those domains. This notation is designed to have an intuitive meaning and yet to have a well defined semantics in terms of the underlying representation.

4. A mechanism which, given a domain description, can plan and perform actions as an agent in the world, interacting with other agents.

Each of these parts builds on those which come before. To avoid, as far as possible, building unwarranted assumptions into the later parts of the work, the early parts are designed to be far more general than is absolutely necessary.

A program called Riple has been written which implements this model of language use. Riple is in fact a fairly general planning system, based to a large extent on previous work on planning and action in artificial intelligence. Its extensions to cope with the problems of text planning take the form of a rather more complex representation for states of the world than is common in planning systems and a simple model of motivating and predicting the actions of other agents when planning. This program is intended to provide evidence that the model presented here is at least practicable. No attempt has been made to produce a useful or efficient system, rather the intention has been to keep the implementation as close as possible to the theory.

§3 The Structure of This Thesis

This thesis is structured to reflect the structure of the work described above.

Chapters 2, 3 and 4 discuss previous work on the structure of text, planning and text generation in linguistics and artificial intelligence. Chapter 2 examines previous work on text structure and purpose in these two fields and from these extracts a number of features of natural language texts which later work will attempt to model. It then presents the model of text structure and interpretation. Chapter 3 examines previous work on planning and rational action to motivate the details of the Riple system. Chapter 4 examines other work on language generation within artificial intelligence, firstly to delimit the scope of the problem we are to tackle and secondly to relate the current work to the current state of the art.

Chapter 5 presents an informal overview of the representation scheme and the mechanisms which we propose to manipulate that representation.

Chapter 6 fills in more details of the Riple system. In particular this chapter describes the language which is used to define Riple planning domains and gives details of the choices made in creating a concrete implementation of the general mechanisms of chapter 5.

Chapter 7 describes the application of the structures which have been defined earlier in the thesis to the problem of text planning. A series of models of text interpretation of increasing complexity are described and for each we describe how it can be used as an audience model to guide the text planning process and how the increasing complexity of the models is reflected in increased sophistication in the texts planned. Each of these models is presented in the notation introduced in chapter 6 and the behaviour of Riple when using these models is described.

Finally, in chapter 8 we discuss the model, its strengths and its limitations and suggest possible directions for further work.

Appendix A formally defines the representation used internally by Riple. Appendix B is a formal definition of the domain description language. Appendix C formally defines the planning process.

Chapter 2

Discourse, Language Use and The Audience

§1 Introduction

In any work on a computational model of discourse planning it would be foolish to ignore the large amount of work on the structure and function of texts which has previously been done in linguistics and cognitive science. In this chapter we examine some of this work and use it to develop a model of language use and discourse structure which can form the basis for a discourse planning model.

§2 The Need For Discourse Structure

In all of this work we shall assume that a discourse can analysed into a sequence of single "utterances". For a spoken discourse, an "utterance" might be a sequence of words made at one time; for a written discourse an "utterance" might be a sentence. In either case we shall assume that an utterance corresponds, roughly, to a (possibly complex) clause.

There are two reasons for believing that there is a need for a level of description between single utterances and complete discourses —

- Usually there are several distinct sub-parts to the goal which a discourse is to accomplish. In such a case we might wish to break down the overall

discourse into pieces each of which relates to the achievement of one or more of these sub-goals.

- Human discourse exhibits structure. Analysis of human produced discourse has shown that there are definite divisions in the text which affect such things as referring expressions [Dale 1986] and 'focus' [Sidner 1983]. A system which plans discourse to be understood by a human audience must produce similar structure and effects if it is to be understood correctly.

Both of these reasons apply recursively. Just as a discourse may have segments corresponding to sub-goals, a discourse segment may be further sub-divided to reflect further levels of sub-goals. Similarly the segmentation in human texts often forms a tree like structure.

Structure of this kind has at least two major effects which a text planner must be able to plan for and take advantage of —

- Structure limits search. The structure of a text provides the audience with guidance in their search for interpretations of new utterances. Much work on text structure has examined the restrictions which structure places on referring expressions (for instance [Dale 1986], [Sidner 1983]) and similar effects can be seen on the search for the meaning of ambiguous texts. Consider the following texts —

  1a) John was very angry.

  1b) So he drove home as fast as he could.

  1c) He hit Bill who had to be taken to hospital.

  2a) John was very angry.

  2b) So he drove home as fast as he could

  2c) and hit Bill who had to be taken to hospital.

  In 1a–1c it seems most natural to interpret the third utterance as a consequence of the second and so interpret "hit" to mean "run over". In 2a–2c where the third utterance is explicitly connected to the second the natural interpretation is that they are both consequences of John's being angry and so "hit" is interpreted as "strike".

For the speaker this kind of search limiting effect is useful; it often means that an otherwise ambiguous phrase or reference can be used without the risk that the audience might misinterpret what is being said.

- Structure conveys information. Often the structure of the text will, itself, convey information beyond that conveyed in the literal meanings of the utterances it contains. If two utterances are placed together in a segment of the discourse then the audience will assume that they are in some way related. For example consider the following extract from [Golding 1980] quoted in [Brown and Yule 1983] —

  > Between where I stood by the rail and the lobby was but a few yards, yet I was drenched before I got under cover. I disrobed as far as decency permits, then sat at this letter but not a little shaken.

  It seem natural to assume that the narrator disrobed because he was wet and that it is this which causes him to feel shaken, yet none of this is explicitly stated. A speaker who did not take into account this kind of audience inference would produce poor text indeed —

  > Between where I stood by the rail and the lobby was but a few yards, yet I was drenched before I got under cover. Since I was drenched, I disrobed as far as decency permits, then sat at this letter but not a little shaken by my being drenched.

  Similarly by closing one discourse segment and starting another the speaker might be taken to be implying that the information conveyed in the two segments, though both related to the superior concept being discussed, are not directly related to one another.

  In many cases the information conveyed by the structure of the text is the speaker's intentions. Often the form of a text provides enough evidence for the audience to determine the goals which the speaker is trying to fulfill. A persuasive argument is structured very differently to a set of instructions or a story. A good discussion of the relationship between text structure and the 'intentional' structure of a discourse is [Grosz and Sidner 1986].

The model presented in this thesis, with its close link between text planning and audience modelling, is designed to take account of these kinds of effect.

§3 Local and Global Focus

Human discourse is not simply constructed of disconnected segments, each fulfilling some sub-goal, strung together at random. There are well defined patterns in how pieces of a discourse, single utterances or sub-discourses, are are ordered. Two important types of pattern in the structure of discourse have been identified in the literature and both have, confusingly, been referred to as "focus". Here we shall follow [McKeown 1982] by referring to them as "local" and "global" focus.

§3.1 Local Focus

The concept of local focus is an abstract description of the way different 'topics' in a discourse (entities, properties, relationships and so on) change in prominence from utterance to utterance. Theories of local focus, for instance [Sidner 1983], [Reichman 1985], specify two things —

- How do different choices of surface form affect the prominence of different topics.
- How does the prominence of topics affect linguistic choices.

An example of a linguistic choice which influences and is influenced by local focus is the choice of the 'topic' item of an utterance. At any point in a discourse certain 'things' are more likely to become the topic of the next utterance and making something the topic of an utterance will increase its prominence. Similarly the use of pronouns depends on the local prominence of the entity being referred to.

The problems of local focus have not been tackled in the work reported here. In chapter 8 we discuss briefly how the mechanisms we have developed might be extended to cope with local focus and reference.

§3.2 Global Focus

Global focus theories describe the way that the large scale information structure of a discourse may be organised. Certain types of discourse, for instance descriptions, will usually contain certain types of sub-discourse, for instance definitions.

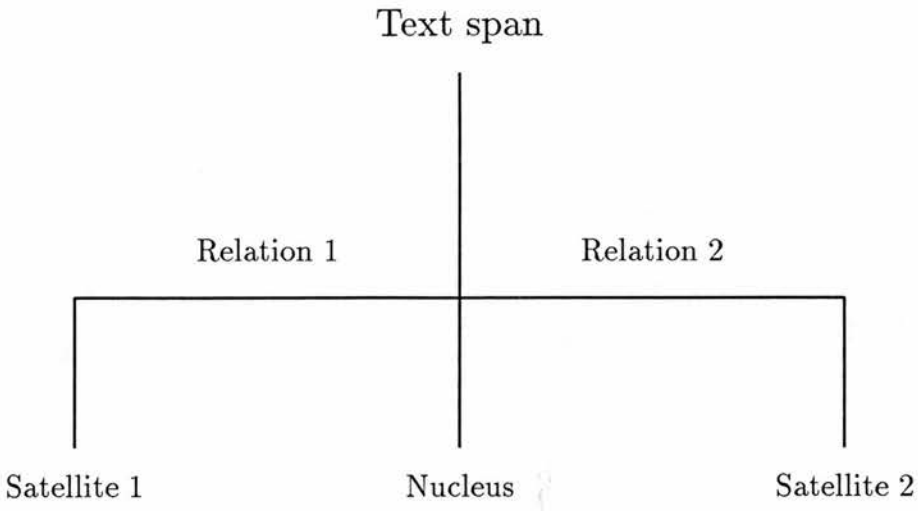


Figure 2–1: An RST schema

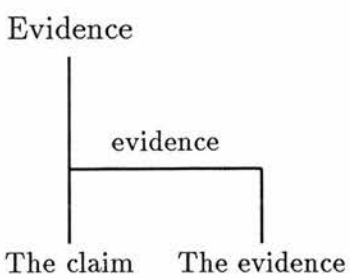
The information contents of these sub-discourses are often related to that of the main discourse in predictable ways; for instance a description will not contain a definition of a term which it does not use. Global focus theories describe the way that the information content of a discourse is distributed in its various parts. The model described in [Grosz 1977] describes the structure and information distribution of two person, task driven discourses. In these discourses the structure of the discourse and the information content are both strongly constrained by the task being performed.

Rhetorical Structure Theory (RST) is a model of text structure and global focus proposed by Mann et.al. [Mann 1984].The basic unit of an RST description of text structure is the schema. A schema describes the relationship between a stretch of text and its sub parts and is usually presented diagrammatically as in figure 2–1. A schema describes the division of a span of text into a number of parts, a 'Nucleus', always shown directly below the name of the schema and one or more 'Satellite'. The order of the parts in the schema gives the most common ordering of the parts in an actual text. Each Satellite is connected to the nucleus by a named 'relation'.

Associated with each relation are four pieces of information which describe how the schema is used.

1. Constraints on the Nucleus.

2. Constraints on the Satellite.



1. The reader, possibly, does not believe the claim.
2. The reader either already believes the evidence or will find it credible.
3. Comprehending the evidence will increase the reader's belief in the claim.
4. The readers belief in the claim is increased.

Figure 2–2: The "Evidence" schema

3. Constraints on the combination of the Nucleus and the Satellite.

4. The Effect.

Hence an RST schema describes both the structure of a text and the way in which the information in a block of text is distributed between its sub-parts.

When RST is used as a descriptive theory, the four types of information described above are used by the person analysing a text to decide which relationships could plausibly be said to hold between two spans of text. For this purpose it is possible to express the constraints and the effect in English and rely on the human analyst to apply this information to the text in hand. Figure 2–2 shows the "evidence" RST schema.

§3.3 Combining Local and Global Focus

Some models have been proposed which describe both local and global focus. [Grosz and Sidner 1986] presents a model where the structure of a discourse is described by three structures

Linguistic The linguistic structure describes the breakdown of the discourse into sub-discourses.

Intentional The intentional structure describes the goals and sub-goals which the discourse segments are intended to achieve.

Attentional The attentional structure describes the way the local focus of the text changes.

The first two structures describe the global focus and the discourse structure, while the last describes the local focus. Similarly [Reichman 1985] presents a model of discourse structure which incorporates notes of the prominence of entities in the discourse.

§3.4 Global Focus and Discourse Planning

Global focus provides an abstract description of the way in which speaker and audience structure their interaction. The linguistic phenomena used to indicate text structure, such as cue words and paragraphing, once again provide a (partial) ordering on possible interpretations which speaker and audience can use to cut down the range of possible interpretations of a discourse. A speaker must indicate the intended structure of a discourse only when the audience might otherwise form an incorrect interpretation; similarly the audience can assume that the most obvious structure of a text is the one intended, since the speaker would otherwise have indicated the correct structure more clearly.

§4 Speaker – Audience Interaction

The most important property of a discourse is its intended effect on its audience. Two major streams of work on this aspect of discourse can be found in the literature.

- Models based on the concept of "speech acts" [Austin 1962], [Searle 1969].
- Models which propose general rules governing interaction and use these rules to explain communicative behaviour; [Grice 1975], [Strawson 1964] [Sperber and Wilson 1986].

In this section we will examine these types of model and then look at what they imply for a computational model of discourse planning.

§4.1 Speech Acts

Speech act models of discourse are based on the assumption that utterances can be divided into classes, each class having well defined conditions of use and effects. As such it might be seen as an extension of the grammatical division of sentences into declarative, imperative and interrogative. [Searle 1969] describes a number of types of speech act and for each gives a number of conditions which must hold before an utterance can be said to be of that type and a list of effects.

The major limitation of speech acts as a model of language use is that, since each type of speech act has well defined sets of conditions and effects, in order to model a significant diversity of utterances in different conditions, targeted at different types of audience we must either have a very large number of speech act types or a small number of very general types.

If we have a large number of types the resulting model is unwieldy and makes few useful generalisations, being essentially a listing of numerous special cases.

If we have a few very general act types, each type lacks the detail needed to be able to model all the distinctions which are important in deciding on the form of a discourse. Also it becomes impossible to determine the surface form of an utterance from the speech act description (or *vice versa*) since not enough information is present.

The difficulty of relating the surface form of an utterance to the speech act being performed led speech act theorists to separate the description of the communicative action being performed from the classification of the surface form of the utterance. The former, often called the illocutionary act or the illocutionary force of the utterance, is used to describe the expected pre-conditions and effects of the action and stands in a many–many relationship to the surface form; that is a given illocutionary action might be performed be any of a number of surface speech acts and similarly a surface speech act might be used, in different circumstances, to realise any of a number of illocutionary acts.

This distinction is a useful descriptive tool since it allows theorists to discuss the restrictions and assumptions which are associated with different types of communicative action without having to discuss that detailed surface form of these actions. However as an explanatory theory, or as the basis of a computational model of linguistic behaviour the distinction is less useful. It is unclear that hav-

ing decided to perform a given illocutionary act we have actually made the task of planning an utterance any simpler.

More recently Cohen, Levesque and Perrault have dome some work on deriving restrictions on speech acts from general assumptions about rationality [Cohen and Levesque 1985], [Perrault 1989].

§4.2 Models of Interaction

Speech act theory can be seen as an attempt to build a model of language usage "bottom up", starting from a classification of utterances into groups and then describing the properties of these groups and the relationships between them. An alternative approach is to start from an examination of the problem of communication and to attempt to describe how human linguistic behaviour approaches this problem.

The central problem faced by human beings in their normal interactions (and by extension by artificial language using systems) is to reliably manipulate the state of another agent whose beliefs, abilities and intentions may be only vaguely known. Human speakers are quite capable of communicating with complete strangers with only the slightest knowledge of their backgrounds. This contrasts sharply with protocols designed for computer–computer communication which usually specify in great detail the structure and behaviour of the systems involved and which often demand that upon initiating communication the systems must exchange information specifying their status. Obviously human language users do sometimes engage in this kind of negotiation, consider the (imaginary) interaction below —

John Do you know Kate?

Mary The police woman?

John Yes. I met her yesterday and...

However usually two people will start a conversation without checking that they share beliefs and yet will usually succeed in communicating. Speech act theories attack this problem by extending the methodology of formal grammar and formal semantics, proposing sets of rules which relate the form and/or semantics of an utterance (and the context in which it is made) to the effects they can be expected

to have. An alternative approach is to propose very general "rules" of communication which, together with models of general reasoning, can be used to explain communicative behaviour.

One very influential model of this kind is that of Grice. In [Grice 1975] he proposes that along with their knowledge of the grammar and semantics of their language, language users share knowledge of conventions of communication which allow them to interpret the utterances of others. The following nine maxims are proposed —

Maxims of quantity

- Make your contributions as informative as is required.
- Do not make your contributions more informative than is required.

Maxims of quality

- Do not say what you believe to be false.
- Do not say that for which you lack adequate evidence.

Maxim of relation

- Be relevant.

Maxims of manner

- Avoid obscurity of expression.
- Avoid ambiguity.
- Be brief.
- Be orderly.

If an audience knows these maxims and assumes that a speaker is following them then they can often use this information to disambiguate an utterance and also to decide on the speaker's intentions in making the utterance.

Conversely, a speaker who assumes that their audience knows they will be conforming to these maxims can make assumptions about how that audience will interpret a given utterance in the current situation, and so decide what utterance to make to achieve a given goal.

This model can explain a wide variety of communicative behaviour. If a speaker is apparently not conforming to the maxims when making a particular utterance the audience will make assumptions which would make the utterance conform.

These assumptions, which Grice calls 'implicatures', can be used to explain how human language users communicate robustly and efficiently without the kind of negotiations to decide on mutual beliefs and intentions described above. Similarly they explain why computer generated texts often seem "wordy" or "pedantic"; a computer text generator without knowledge of such maxims and of their audiences use of them will spell out things which the audience could be expected to assume.

However Grice's maxims are obviously too vague to serve as the basis of a computer model of language use. Some work has been done on formalising subsets of the maxims to serve as a formal model of implicature, e.g. [Gazdar 1979]. Such attempts, though they may capture interesting regularities in the relationship between context and utterance meaning on the one hand and interpretation and implicature on the other, do not really capture the essence of Grice's model which was intended as a general mechanism applicable to any utterance and context.

In [Sperber and Wilson 1986] Sperber and Wilson put forward a model of communication which, building on the insights of Grice, describes in more detail the mechanism by which the expectations of language users guide communicative behaviour. Their 'Relevance' model has four major parts —

- An (informal) model of memory structure and inference.

- A definition of communication as being the action of a communicator making some set of assumptions (including the assumption that the communicator intends to communicate) more 'manifest' to their audience. 'Manifestness' is a concept defined by Sperber and Wilson which may, for the purposes of this discussion, be understood as measure of how aware of an assumption an agent is.

- A restatement of Grice's maxim of relation as follows —

  > Every act of ostensive communication communicates the presumption of its own optimal relevance. *(Page 158)*

  where 'relevance' is defined as follows:

  > *Extent condition 1:* a phenomenon is relevant to an individual to the extent that the contextual effects achieved when it is optimally processed are large.

> *Extent condition 2:* a phenomenon is relevant to an individual to the extent that the effort required to process it optimally is small. *(Page 153)*

- A general rule which an audience may use to decide which of the possible interpretations of an utterance is the one which the speaker intends them to recover:

> At every stage in disambiguation, reference assignment and enrichment, the hearer should choose the solution involving the least effort and should abandon this solution only if it fails to yield an interpretation consistent with the principle of relevance. *(Page 185)*

Though more explicit than the Gricean maxims, these principles are still not detailed enough to form the basis of a computational model of communication without extra assumptions.

The least satisfactory part of the theory is the model of memory and inference. Although Sperber and Wilson go into some detail about the properties which this model should possess, "chunking" of concepts and non-deductive inference for example, these properties are far from being enough to provide a full model. We discuss some more complete representation schemes in section §5 below.

Manifestness is a very complex concept which the current work does not tackle. In developing our model we shall assume that all assumptions and stimuli are either totally manifest to an agent or totally invisible. In particular, we shall assume that all actions are either private, invisible to all other agents, or public, visible and automatically observed by all other agents. Inferences are an example of the former and ostensive communication of the latter; no other types of action are discussed in this thesis.

Sperber and Wilson's principle of relevance is the basis of the assumption which we shall make in chapter 7 that an agent will always attempt to interpret any utterance which is made to them. Any such utterance is implicitly guaranteed to be worth processing by the agent making it.

The final part of the theory, the assumption that an audience is justified in always taking the simplest and quickest path when there is a choice in interpretation, can be seen as a justification for applying standard AI heuristics, such as least commitment, to interpretation and thus for the speaker to assume that the audience will

use them. We shall take a special case of this rule and assume it can be used as a general rule of interaction: in order to ensure that an agent performs a particular action it is only necessary to make sure that the agent has a goal which that action can fulfill and that all "simpler" actions which might also fulfill it are impossible. This rule, of course, assumes that there is a total ordering on the various possible actions which an agent might perform to achieve a goal. In the case of language we assume there is an ordering on the possible interpretations which the audience might ascribe to an utterance. In general such an ordering may not exist, but we assume here that there is always at least a partial ordering which is, to a great extent, shared by all agents. In those cases where there are non-ordered choices available to the audience then it is the speaker's task to modify their utterance to eliminate those choices.

This rule of interaction forms the basis of the concept of "motivation" developed in section §5.4.5 of chapter 5 and described more fully in appendix C.

§4.3 A Model of Interaction for Discourse Planning

From the work described in this section we can put forward a model of communication which forms the basis for the discourse planning model presented in this thesis.

Communication is taken to involve a speaker[1] performing actions which modify the environment of an audience in such a way as to make it clear to the audience that this is a communicative action. The audience will then, because of the implicit guarantee of usefulness which comes with an utterance, attempt to find an interpretation for it. The *first* suitable interpretation found can be assumed to be the intended one since if it was not the speaker would have made that interpretation impossible.

Thus the task of the speaker in planning a text is to determine an ordered set of communicative actions which will cause the audience to construct an interpretation which will have the desired effect. The details of a text, such as ordering and cue

---

[1]In dialogue there will, of course, be more than one speaker and there will be mechanisms to control the alternation of speakers. However this is not discussed in this thesis.

words, must be chosen to ensure that the interpretation which the audience will first construct is the one intended by the speaker.

§5 Representing the Audience's Beliefs

One of the central issues in the creation of a model of linguistic behaviour is that of how the speaker's beliefs and the speaker's beliefs about the audience's beliefs interact to shape the text. There are three major properties of a representation which determine how well it supports text planning.

Power Language use can involve many interacting and subtle types of relationships between different agents' concepts and their views of the world. The ability of a model to represent such relationships is obviously crucial.

Delicacy In addition to modelling complex relationships, a text planner must be able to make fine distinctions. A representation which obscures distinctions between what the agent *believes* to be the case and what they could *infer*, for instance, will be unable to model linguistic behaviour whose purpose is to guide the audience in making inferences.

Transparency Both the desire to produce an understandable model and the practical considerations of implementing the model as a computer system favour representations which make the concepts most important to the text planning process explicit.

These properties are obviously somewhat at odds with each other. A more powerful representation may gloss over some distinctions captured by a less powerful one and a representation capable of making fine distinctions may be harder to interpret than a courser one. In designing a representation we must therefore compromise between power and transparency in order to create a theory which can model interesting linguistic behaviour whilst still being understandable and implementable. We will now examine a few of the many representation schemes which have been proposed for use in modelling language and its relationship with the world and agents' beliefs.

§5.1 Modal Logic and Related Schemes

The most well established representation schemes for beliefs and intentions are the various forms of modal logic ([Hughes and Cresswell 1968]). Such logics usually take the form of a propositional or first order predicate logic enriched with operators denoting belief, knowledge, intent and other propositional attitudes. The interpretation of such a logic is usually provided either by describing it as a formal system with axioms and inference rules or by giving the logic a formal semantics, usually based on the concept of a "possible world". In general these logical representations, though differing in detail share a number of advantages and disadvantages. Modal logics are powerful, having the ability to represent complex issues of the relationships between concepts in different modal concepts by means of interactions between the scopes of quantifiers and modal operators. However in standard modal logics this power is balanced by the fact that it is not easy to determine from a collection of formulæin such a logic just what is or is not the case in the state of the world being described and also by the problem of 'logical omniscience'.

A representation is called logically omniscient if the beliefs of an agent expressed in the representation are closed under logical inference. In such a representation it is impossible to distinguish between an agent being in a position to infer some fact and them having made that inference. This restriction causes obvious problems for modelling certain types of linguistic behaviour such as —

Reminders If an agent knows something then there is no sense in which that belief could be "unavailable"; all beliefs have the same status. The system could never plan to remind someone of a fact which they already believed, an action which occurs very regularly in human generated texts.

Guiding Inference Very often a human will explicitly indicate an inference which is known to be valid by the audience. This is because a human audience will not necessarily make all possible inferences from their beliefs and so the speaker must sometimes explicitly state which inferences should be made. A model of a logically omniscient agent will never do this kind of guidance since it is not capable of representing the fact that an inference has not been made.

These phenomena are both results of limitations in the audience's inference. If the text produced by a generating system is to be easily comprehensible by a human audience, then the system must take these limitations into account in its model of its audience.

Closely related to modal logics is the representation described by Moore in [Moore 1981]. In this paper, Moore presents a representation which uses the concept of possible worlds and relationships between them to directly represent states of the world and agent's beliefs and actions. This kind of model is normally used to give a semantics to modal logics ([Kripke 1963]) but here is used as a representation in its own right. There is a natural relationship between the possible worlds of Moore's representation and the planning concept of a state of the world which is exploited both by Moore's extension of the logic of belief to cover actions and by Appelt's KAMP system (which we discuss in more detail in chapters 3 and 4). The greatest problem with this representation from the point of view of text planning is that, like standard modal logics, it models logically omniscient agents.

In contrast to Moore, Konolige ([Konolige 1982]) presents a formalisation of belief and action which is basically syntactic rather then semantic in nature. Konolige's representation takes the form of a logic (which he calls the meta-language or ML) which is used to talk about theories which agents construct about the world and their relationships. These theories are in turn described in a different logic (the object language, OL). In turn he allows OL to talk about theories in a nested object language OL′. One important difference between Konolige's model and a more standard modal logic is that Konolige explicitly takes what he calls an 'egocentric view' of the representation – that is, a given set of expressions in ML is taken to describe the views of some specific agent, whereas in a modal logic, a collection of sentences is generally taken to describe the world in an objective way. A problem which Konolige has to deal with in order to define a useful representation is that of 'cross-world identity'. Since the representation contains (at least) three nested languages each with its own space of objects, some way must be provided to relate entities described in each language to each other. Konolige chose to do this by providing a syntax for defining standard names for objects which are shared between ML and OL. Konolige also avoids the problem of logical omniscience by representing agents explicit beliefs rather than just *describing* those beliefs.

Other logical solutions to the problem of logical omniscience are presented by

[Levesque 1984] and by [Fagin and Halpern 1988]. Levesque develops a logic which distinguishes an agent's explicit beliefs from those things which they might logically infer from them which he calls implicit beliefs. Fagin and Halpern extend this logic to cope with higher level beliefs and beliefs of other agents. They also present two further approaches to avoiding logical omniscience. One is to limit the set of formulæof which an agent is 'aware', allowing the agent to believe only such formulæ. The second is to divide up an agent's belief into clusters and only allow inference within such a cluster. These representations seem to capture intuitions about what we understand by 'belief' somewhat better than does standard modal logic.

§5.2 Non Logical Representations

Situation semantics is a representation framework created by Barwise and Perry ([Barwise and Perry 1983]) and subsequently used to represent many types of linguistic phenomena. The main representational object in situation semantics is the situation type which for our purposes can be seen as an assignment of truth values to a set of atomic statements about the world built from entities and relationships. For instance —

in s: hungry, Jackie; yes
sleeping, Jackie; no
sleeping Molly; yes
hungry, Molly; undefined

This kind of representation has the advantage of being able to represent limited information, though as it stands it is obviously very limited in power. Barwise and Perry extend the representation by allowing 'indeterminates' and 'roles' (typed indeterminates) which are similar to the variables in a logic, allowing generalisations to be captured. Beliefs can then be represented as relationships between agents and situation types defined in terms of roles. This representation allows generalisations across beliefs and also can represent exactly what has or has not been inferred by an agent at a given point in time.

In [Fauconnier 1985] Fauconnier presents a type of representation which he calls 'mental spaces'. The most important feature of this representation is the fact that the relationships between the spaces representing, for instance, the beliefs of various agents are not simple connections between the states, but include mappings between the entities mentioned in the spaces. This facility allows the mental

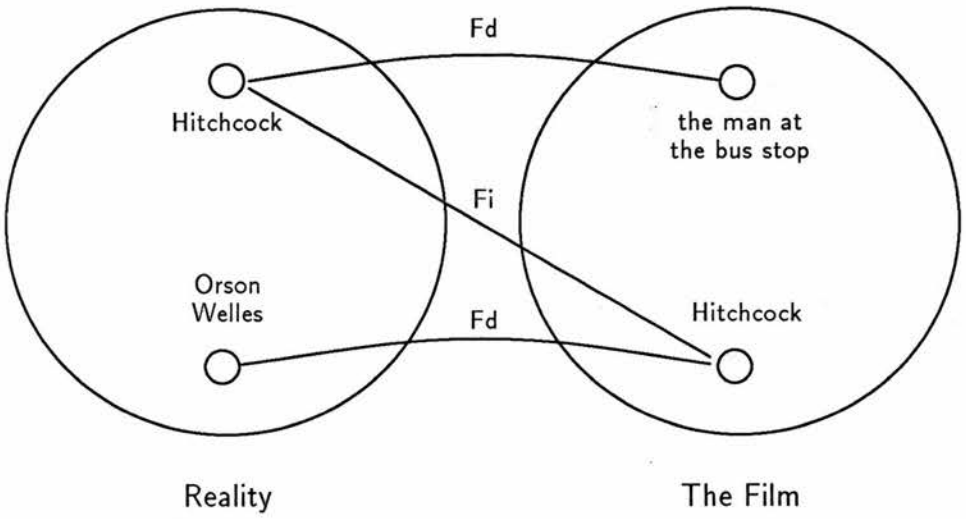


Figure 2–3: An Example of a Pair of Mental Spaces

spaces representation to be used to model situations in which there is confusion between agents as to the identity of individuals. For example, consider a situation where a film is made about the life of Alfred Hitchcock. In the film stars Orson Welles as Hitchcock and Hitchcock himself plays a small part as a man at a bus stop. Figure 2–3, taken from [Fauconnier 1985], page 36, shows this situation diagrammatically. There are two 'spaces' each with two entities. There are also two separate relationships between the entities in the spaces, Fd, relating actors to rôles and Fi relating people to their representation in the film.

Another nested belief representation has been proposed by Wilks, [Wilks and Bien 1983], [Wilks and Ballim 1987]. This representation is unusual because it does not store nested belief contexts in a hierarchical manner, rather the system maintains models of the beliefs of various agents and when it needs to consider nested beliefs it constructs the relevant nested structure from what it knows about the agents involved.

§5.3 What do we need from a Representation for Text Planning?

The representation schemes we have examined above present us with various parts from which to construct a representation tailored to the needs of text planning. For reasons which will be presented in more detail in chapter 5 we make the following choices —

In order to avoid problems of logical omniscience and to allow the text planning process to model the audience's expected reasoning when interpreting the text we use a representation which explicitly lists facts which are true in different belief sets and rely on explicit modelling of inference in the planning process to capture the active nature of the audience. We will also limit our audience models to reasoning within the context of a discourse.

Like Konolige we represent nested beliefs by having the beliefs of one agent contain descriptions of the beliefs of others. Unlike Konolige our representation is a simple conjunction of terms, rather like a situation type in situation semantics.

Cross-world identity is to be handled by mappings, similar to those proposed by Fauconnier attached to every belief set.

§6 'Points', a Simple Model of Text Structure for Text Planning

Now that we have examined some of the existing work in linguistics and knowledge representation which is relevant to problems of text planning, we are in a position to describe the simple model of text structure which forms the basis of many of the choices described in later parts of the thesis. Although it is presented here in isolation, it should be remembered that this model and the planning and reasoning mechanisms to be described later have developed together. Some of the limitations of the 'points' model reflect limitations of the planning model and many of the decisions taken in developing the planning system were motivated by the need to be able to describe interpretation models such as this one within the system.

The overall form of this model is similar to that of a number of models of text structure proposed within computational linguistics, for instance that of Grosz and Sidner [Grosz and Sidner 1986] or Reichman [Reichman 1985]. However we do not attempt to provide a model which provides wide linguistic coverage, rather the model presented here is designed specifically to highlight issues of audience modelling. For this reason we concentrate on a few features of human language usage which seem to depend heavily on the speakers knowledge of the audience —

Reminders Reminders are essentially attempts to manipulate the audience's focus of attention.

Focus Shift The global focus structure of a text (in the sense of section §3 above) can be seen as a reflection of the speaker's attempt to control the audience's centre of attention over the course of the text. Knowing where to change focus and how to signal the change is another area of text planning requiring a great deal of knowledge of the audience.

Cancelling Inferences It is as important that the speaker know what the audience is likely to infer from a text as it is for them to know what the text literally means to the audience. Often it is necessary for the speaker to add to a text utterances whose purpose is to block incorrect inferences by the audience. Deciding when this need be done requires the speaker to predict the audiences interpretation of the text at a fairly detailed level.

Disambiguating Utterances Human produced texts are often ambiguous when examined in isolation and yet perfectly understandable to the intended audience. This happens because the speaker when constructing a text can predict how the audience is likely to understand it and, conversely, the audience knows that the speaker will have been able to make such a prediction and so can be reasonably sure that the interpretation they reach will be the intended one. For this to work the speaker must be able to add utterances (or perhaps just extra information in referring expressions) to a text to block unwanted interpretations.

In the following subsections we shall present the model informally, first describing what a discourse structure looks like, then how the structure is built and how the position of an utterance in the structure is determined. Next we present rules for interpreting utterances within a discourse and discourse units within their surrounding contexts. Finally some examples of interpretation using this model are presented. These examples will be used in section §9 of chapter 7 as examples of text planning.

§6.1 Representing Utterances

Before we define our model of discourse structure and interpretation it is necessary to adopt a language for describing the utterances which form the input to the interpretation process and, conversely, the output from the text planning process. In chapter 4 we will examine work which has been done on the problem of generating surface forms from abstract descriptions of utterances. We will see that the input to such tactical generation systems was usually at a level of abstraction comparable to an annotated case frame and that is the level we will assume is reasonable in this interpretation model and others.

Thus our utterances will be represented as case frames, similar to those used in earlier sections, with the addition of any of the following markers —

subjunctive This indicates that the utterance is being used to convey a fact which is to be taken as an assumption or a condition of some kind. A typical surface realisation might be to make the utterance the 'if' part of an 'if...then' sentence structure.

conclusive This indicates that the utterance is marked as a conclusion of some kind. A typical surface realisation would be the presence of the cue word 'so'.

contrastive An utterance would be marked as contrastive if it was being placed in contrast to preceding text, for instance to block an otherwise likely inference by the audience. Contrastiveness might be marked in an utterances by including the cue word 'but' or by using contrastive stress or intonation.

Obviously these marks do not represent an interesting theory of pragmatically significant features of spoken or written language. They have been selected simply to make the distinctions necessary for the examples to be presented below.

§6.2 Discourse Structure

In common with many other models of text structure (e.g. [Grosz and Sidner 1986], [Reichman 1985], [Kamp 1984], [Dale 1989]), we will represent the structure of a

text as a tree of contexts. The leaves of this tree will be trivial discourse contexts representing individual utterances. Each context will have associated with it the following information.

- A "description" which is a representation of the state of the world being assumed at that point in the discourse. This corresponds to the "discourse context" of earlier models.
- A subset of the description called the "assumptions".
- A subset of the description called the "effects".
- The position of this context in the structure of the text.
- A "point"; that is a reason for this context being part of the text.
- A set of "markers" giving any information about the point of this context which is known.

At any stage a discourse context is either open or closed. If it is open it may still be extended, if closed it is finished and must have been interpreted and its effects will have been inserted into the surrounding discourse.

The audience's task when interpreting an utterance will be to determine this information and then use it to interpret the discourse context of which this utterance is a part (and so on up the tree). When the 'point' of the entire text is known the audience will be in a position determine what action is relevant (for instance accepting a fact).

Here we shall be interested in only four kinds of 'point' —

reminder A discourse context has the point 'reminder' if it is intended to make more salient some piece of information which is already in the surrounding context.

inference A discourse context has this point if it is intended to cause the audience to draw an inference from information already in the surrounding context. We shall also deal with 'conditional' inferences, where the inference is of the form

if "..." were true then we could infer "...".

conclusion A 'conclusion' is assumed to be like an inference except that it is marked in some way as being important to the surrounding discourse. Thus while "Mary owns a car." might be a reasonable utterance to find in a discourse, we would expect later parts of the discourse to make use of it in some way, whereas "So, Mary owns a car" might be an end in itself. The latter would be marked as a 'conclusion'.

assumption A discourse context which has the point 'assumption' is intended to communicate the assumptions under which the surrounding context is to be interpreted.

correction A discourse context which has the point 'correction' is intended to correct an expected mistaken inference by the audience.

Informally, "surrounding context" can be taken to mean the discourse context of which this context is a part or the one containing that and so on with the agent's beliefs forming the outermost context. A more formal definition of "surrounding context" will be presented in section §6.4 when we give more precise characterisations of each kind of point.

The surface form of utterances and how they are created from or interpreted as abstract descriptions such as we are discussing here is complex. For instance a conclusion might be realised without any cue word such as 'so', especially in informal spoken contexts. A pause or a marked intonation contour might be enough. When we give examples later and in chapter 7 we will give what we hope are plausible renderings of the utterances planned or interpreted.

§6.3 The Interpretation Process

As the speaker produces utterances, the audience must insert them into the discourse structure which they are creating. In order to complete our model we therefore need to describe how the audience chooses the discourse context in which to interpret a given utterance and how that utterance is placed into the discourse structure.

We assume that the surface form of the utterance has been analysed to produce a semantics of some kind and that this analysis gives sufficient information to be

the contents of a discourse context. Thus as each utterance is received a discourse context representing that utterance is inserted into the discourse structure being created.

When a discourse is started it is represented by an empty discourse context; that is one whose 'description', 'assumptions' and 'effects' are empty. This becomes the 'current' context.

As each utterance in a discourse is interpreted, the discourse context corresponding to that utterance is inserted as a child of the 'current' context. The interpretation of this discourse context must take place before its parent context can be interpreted. If this is not possible (because there is no reasonable interpretation for this utterance in the current context) the current context is closed and the new context is inserted as a child of the current context's parent, if possible. In this way an utterance can indicate the end of any number of discourse structures as well as carrying its own content. This process is shown diagrammatically in figure 2–4.

When the end of a discourse is reached,[2] the audience must close all the contexts which remain open, starting from the bottom of the tree and working upwards. In effect, the end of the discourse behaves as an utterance which can be interpreted in no context. In doing this we will arrive at an interpretation for the discourse context at the top of the tree, that introduced by the first utterance of the discourse, and thus at an interpretation for the discourse as a whole. This top level discourse context must be interpreted with respect to some surrounding context. For the purposes of this work we can assume that the audience's beliefs form the context for interpreting the entire discourse[3].

This is of course a great simplification of how real discourse works. Most importantly perhaps, the interaction of discourse with more general activity and social

---

[2]We assume that the audience can determine when the text ends. This assumption is reasonable for most written texts, though problematic for spoken discourse.

[3]Another alternative might be for the audience to use their model of the speaker's model of the audience's beliefs as the basis for interpreting the discourse. This would have the advantage of giving a better model of the effects of the audience's knowledge of the speaker on interpretation, but the extra complexity would obscure the presentation here.

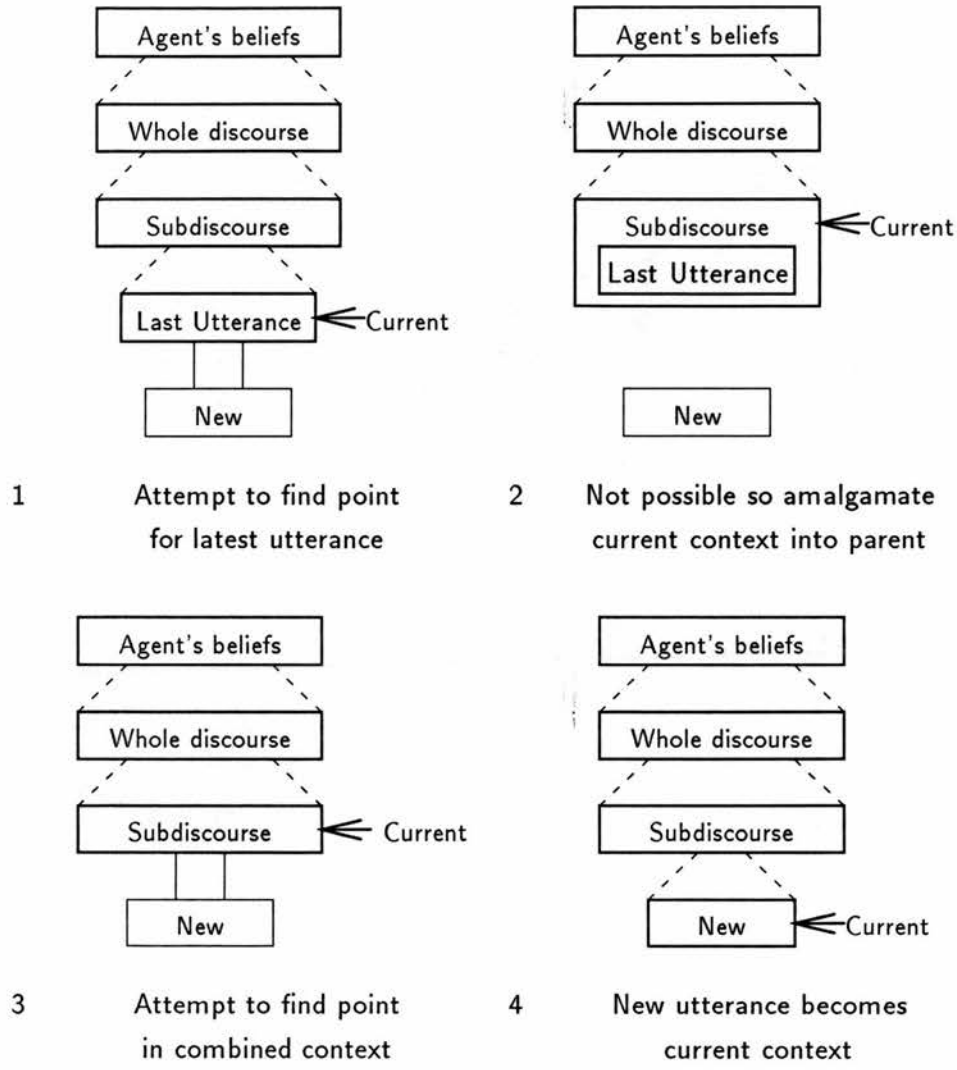


Figure 2–4: How Adding a new Utterance Might Close Currently Active Discourse Contexts

---

context is very complex. For instance reminding someone that they promised to do something might cause them to do it or to apologise or just to laugh depending on who the speaker is. However in this work we shall ignore these issues.

§6.4 Determining The Point

The following rules are used to interpret a closed discourse context. In all cases 'current context' means the last open context while 'new context' means the closed context being interpreted. The 'surrounding context' is the parent of the current context.

None If the effects of the new context are already in the current context then the new context has no point in the current context.

Reminder If the effects of the new context are already in the surrounding context but not in the current context and the assumptions of the new context are plausible in the current context then mark the new context as a reminder.

Conclusion If the new context has the mark 'conclusive' and the effects of the new context are 'plausible' in a context consisting of the description of the current context and the assumptions of the new context then mark the new context as a conclusion and add the facts inferred in proving 'plausibility' to the description and effects of the current context.

Inference Otherwise, if the effects of the new context are 'plausible' in a context consisting of the description of the current context and the assumptions of the new context then mark the new context as an inference and add the facts inferred in proving 'plausibility' to the description and effects of the current context.

Assumption If the new context has the mark 'subjunctive' then mark the new context as an assumption.

Correction If the new context has the mark 'contrasting', the negation of the effect is in the current context and the effect is plausible in the surrounding context then mark the new context as a correction.

If none of the above cases hold then the new discourse context cannot be interpreted in the current context. In this case the current context must be closed and the new context interpreted in the next higher context as described in section §6.3.

The notion of plausibility we shall assume is very simple. A fact is plausible if it is part of the world model of the current context, if it is an assumption of the current context or if it is inferable from the current context. We allow the audience to add any fact to the assumptions of a context as necessary to get an interpretation.

It is not clear how the audience can determine except by exhaustive search which assumptions are needed to make a discourse interpretable. However since we shall be using this model only as a guide to generation, the actual method the audience might use is not significant; all that a text planner must know is which assumptions the audience must make to interpret a fragment of the discourse so that it can plan to block them and/or supply those which it requires them to make.

Notice that the facts inferred when determining something is plausible are added to the description of the context in which the inference is done and also to the effects. We do this to make the audience slightly more awkward to plan for than it would be if inferred facts were local to a discourse context. This is another place where the precise interpretation rules would depend on the type of discourse. An alternative is to give inference rules two classes of consequent, important ones which go into the effects of the context and background which go into the world model.

§6.5 Closing and Interpreting contexts

Closing a context must force it to have its intended effect on the surrounding context. Since the context being closed is a complex object which has been modified by subsidiary contexts while it has been open, we must examine the context to see what the point of the whole section of text which it represents might have been. Thus, the first thing which must be done when closing a context is to apply the rules of section §6.4 to determine the point. [4]

Once the "point" of a context is determined we can close the context and modify its parent to reflect its effects. The information associated with a discourse context is used to decide how its surrounding context must be extended.

1. If the closed context has point 'assumption' then add its effects to the assumptions of the parent context.

---

[4]This will, of course mean that single utterance discourse contexts will have their points determined twice, when they are inserted into the structure and then when they are closed. Though notionally separate these two determinations will be identical and we will avoid them when applying the model in section §8 of chapter 7 .

2. If the closed context has type 'conclusion' and all the facts in its assumption set are plausible, add its effects to the description and the effects of the parent context.

3. Otherwise, if the closed context has point 'conclusion' then add an inference rule to the description and effects of the parent context with the closed context's assumptions as the antecedent and its effects as the consequent.

4. If the closed context has type 'inference' and all the facts in its assumption set are plausible, add its effects to the description of the parent context.

5. Otherwise, if the closed context has point 'inference' then add an inference rule to the description of the parent context with the closed context's assumptions as the antecedent and its effects as the consequent.

6. If the closed context has point 'correction' then add the effects to the description of the parent context.

7. If the closed context has point 'reminder' then add the effects to the description of the parent context.

The difference between 'inference' and 'conclusion' contexts comes out here. Since inference contexts do not affect the effects of the 'current' context they will play no direct role in the interpretation of that context. They can, however provide information needed for later conclusions.

§6.6 Summary of Model

In summary, the following steps must be performed when interpreting an utterance —

1. A discourse context corresponding to the utterance is constructed by syntactic/semantic processes. The precise nature of these processes is not important to us except in terms of their output. In the following descriptions we will make some assumptions about this output, but in general we will ignore these lower level processes.

2. The "point" of the context is determined using the rules of section §6.4. If no point can be found, the current context will be closed and its parent becomes the current context. This is repeated until a "point" is found.

3. The new context becomes the child of the current context and is made the current context for later utterances.

4. At some point the context will need to be closed, either because the end of the discourse has been reached or because a latter utterance does not have a "point" within it. The rules of section §6.4 are applied again to determine the point of the sub-discourse which the context now represents.

5. The rules of section §6.5 are applied to determine this context's effects on its parent.

6. The parent of this context becomes current.

This process will, in general, give rise to a very deep context tree with each utterance interpreted as a sub-context of the preceding one.

§6.7 Simple Examples of Interpretation

In this section we will examine simple discourse fragments and their evaluation according to the above rules. These fragments will be examined again in section §8 of chapter 7 as examples of the text *planning* process using this interpretation model to model the audience.

§6.7.1 A Trivial Example

Our first fragment is simply —

3) Mary owns the blue car.

Assume that the audience's beliefs already contains the facts `owns(mary77,-car45)` , `is_car(car45)` and `is_blue(car45)`. Notice that in such a situation the utterance is pointless, but we use it as an example here because it is very simple and shall build upon in later examples. We can expect the interpretation to fail to find a reasonable 'point' to this utterance.

The discourse context constructed to represent this utterance would be something like that in figure 2–5.

In this representation we make some assumptions about the relationship between utterances and their representation. Specifically we have assumed that the input

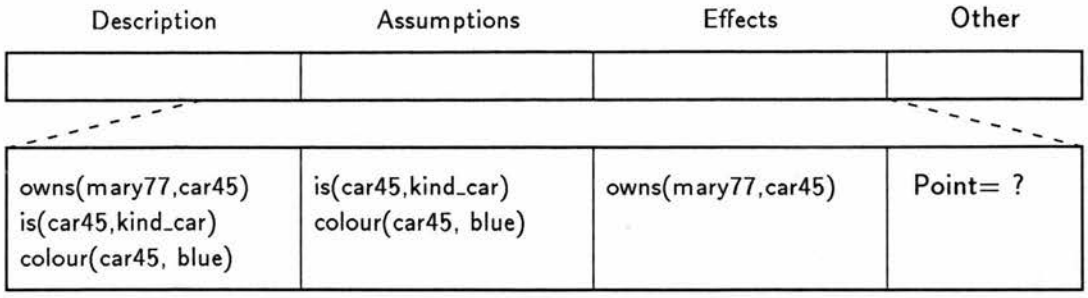


Figure 2–5: The Initial Discourse Context for Example 3

process places facts used in evaluating the referring expressions in the expression into the assumptions of the utterance's representation. This might be the result of some kind of reference resolution scheme. From the speakers point of view, this information will be determined by a reference planning system such as those developed by Appelt [Appelt 1988] and Dale [Dale 1989].

Since 3 is the first utterance in the discourse, its description becomes the child of the empty context which represents the discourse as a whole and it becomes the current context. We have now reached the end of the discourse so this context must be closed and interpreted.

Applying the rules from section §6.4 we find that since the effects of the context are already in the audience's beliefs, which play the role of surrounding context (i.e. parent of the current context) here, but are not in the current context (the current context being empty), 3 must be a reminder. We add it to the base of the chain of contexts and make it the current context.

The end of the discourse now causes us to close the current context. There have been no sub-discourses and so the point of the context will remain unchanged. Applying the rules for extending the parent context to the newly interpreted context, we add the fact `owns(mary77,car45)` to the description of the parent context.

We must now close the top level context of the discourse. The point determination rules provide no "point" for this context and since there is no parent context to close we can not move up the tree and so the interpretation of the discourse must

fail, as we expected. The utterance is a perfectly reasonable reminder, but a reminder on its own does not form a valid discourse in this model.[5]

§6.7.2 An Example of Inference

Consider the following fragment –

4a) If John is a police officer

4b) he is tall.

4c) So, he is a good choice for the basketball team.

We assume here that each clause of a complex sentence such as "If John is a police officer, he is tall" is represented as a separate utterance for interpretation purposes. Lower level interpretation with which we are not concerned here might treat these as one utterance and build a discourse context for the whole sentence. We choose this analysis for an example of interpretation because it presents an example of an embedded discourse context in a simple text.

Assume that lower level interpretation processes produce representations for these three utterances as shown in figure 2–6, and that the audience's beliefs contain inference rules making being tall be plausible if someone is a police officer and being good for a basketball team be plausible if someone is tall.[6] The audience might interpret these utterances as follows —

1. At the start of the discourse, the audience will create an empty discourse context to represent the whole following discourse.

---

[5]For a reminder to be a reasonable discourse on its own, the underlying representation would have to have some notion of long term 'salience', which it does not.

[6]In these diagrams we have included some information which might be expected to have come from lower level interpretation processes, for instance the fact that John is male. This has been done simply to indicate what kind of information might be produced in this way. Since this information is not involved in the interpretation of the utterances we shall omit it from later discussion.

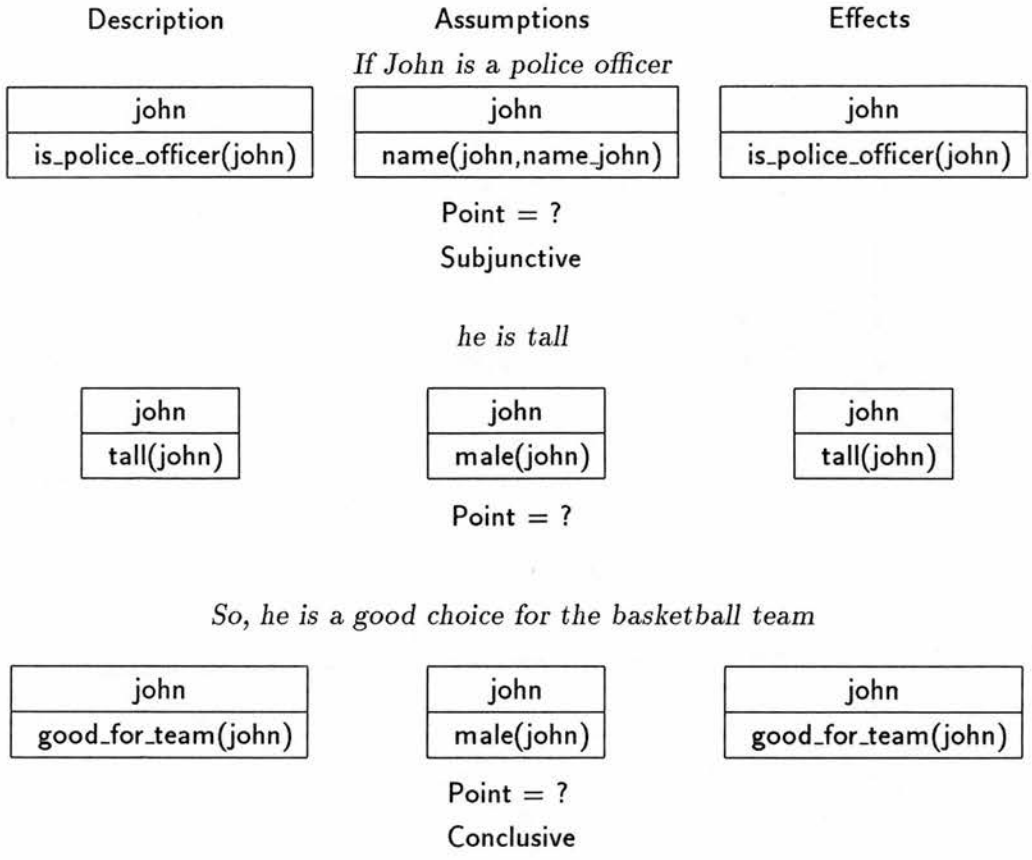


Figure 2–6: The Representations for Utterances 4a, 4b and 4c

2. The discourse context corresponding to utterance 4a will be positioned as a child of the top level context.

3. Applying the rules from section §6.4 the audience will find that this context should have point "assumption", since it is marked "subjunctive".

4. The audience will now attempt to interpret utterance 4b. The "current" context for this interpretation will be the context created by interpreting utterance 4a.

5. Once again applying the rules of section §6.4 the audience will determine that this utterance has point "conclusion" since we have assumed they have an inference rule indicating the plausibility of a police officer being tall.

6. When trying to find the "point" of utterance 4c the system will fail since the description of its parent context, that corresponding to utterance 4b, is empty. Thus we must close the parent.

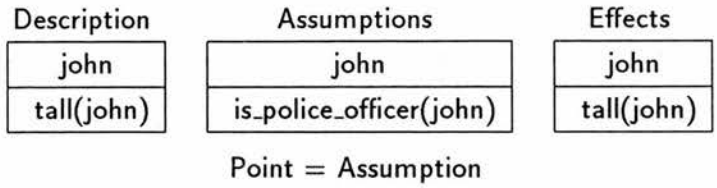


Figure 2–7: The Current Discourse Context after Interpreting 4b

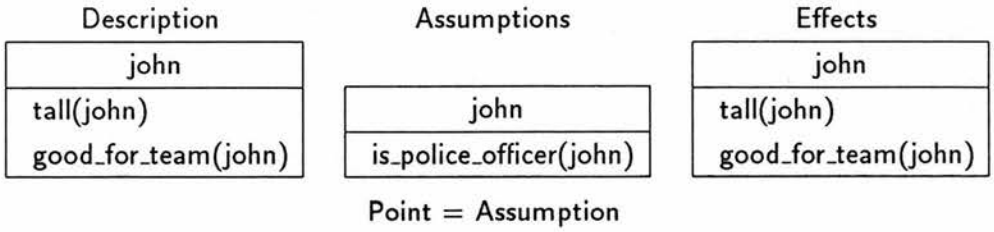


Figure 2–8: The Current Discourse Context after Interpreting 4c

7. Since the discourse context for 4b has no sub-contexts its point remains unchanged. Applying the rules of section §6.5, since the assumption set is empty, we add the effects (just the fact `tall(john)`) to the effects and description of its parent. The current discourse context now becomes that shown in figure 2–7.

8. We now try once more to determine the point of 4c. In this context it is determined to be a conclusion, it is plausible given that John is tall and marked as 'conclusive'.

9. We have now reached the end of the discourse and so we must close all of the discourse contexts which remain open. First that corresponding to 4c; its point remains unchanged as "conclusion" so we add its effects to the surrounding context.

10. To close this context we must re-determine its point. The sub-contexts have changed its context considerably and the rules of section §6.5 will assign the point "inference" to it, giving the context shown in figure 2–8.

We now have to close the final context. To do this we shall need refer to the audience's beliefs to determine what is plausible. So far we have made no assumptions about these beliefs apart from the presence of the two inference rules we have used. How the discourse as a whole will be interpreted will depend on whether

the audience actually believes that John is a police officer. We consider both cases below.

11a If `is_police_officer(john)` is plausible to the audience, the rules of section §6.5 will cause the effects of the final context, `tall(john)`, `good_for_team(john)` to be added to the audience's beliefs.

11b If `is_police_officer(john)` is not plausible in the audience's beliefs then a new inference rule will be added to the audiences beliefs with antecedent `is_police_officer(john)` and consequent `tall(john)`, `good_for_team(john)`.

§6.7.3 An Example of a Correction

A rather more interesting example of speaker – audience interaction can be seen when the speaker finds it necessary to correct the audience, believing they have made an incorrect inference. For example —

5a) Kate is a police officer,

5b) but she is not tall.

If we assume that the audience has a stereotypical image of a police officer which includes the fact that police officers are usually tall, then the natural way of interpreting the above text is as follows (skipping many of the details which are similar to the previous examples) —

1. A context is created for the first utterance. If we assume that it is plausible that Kate is a police officer then the rules for determining the point will mark this as an inference. The world model and effects of the context will now contain the information generated in proving plausibility. We assume this includes Kate being tall as well as a police officer.

2. Closing the inference context will update the world model of the surrounding discourse to include these two facts.

3. Now a context will be created for the second utterance. This utterance will be marked as contrastive and, since the negation of its effect is indeed in the parent context the point of the new context will be set to be 'correction'.

4. This context can now be closed, resulting in the cancellation of the inference that Kate is tall.

§6.7.4 A Longer Discourse

Here is a final example text (indented to show a plausible structure).

6a) John is in the shop.

6b) If he has money

6c) then he can buy some food.

6d) He has his wallet.

6e) So he has money.

6f) So he can buy some.

We assume that the audience has inference rules which can be paraphrased as:

> If someone is in a shop and they have money then they can buy food.
> If someone has their wallet then they have money.

We also assume that John being in the shop is asserted in the surrounding context and so is the fact that he has his wallet.

Interpretation would go as follows —

1. The first utterance is obviously a reminder, given our assumptions. It's interpretation will result in the creation of a new context with a world model asserting that John is in the shop.

2. The second utterance is interpreted in this context. It is marked subjunctive and so creates a context with point assumption.

3. The third utterance is now interpreted giving a context with point conclusion. In order to do this the audience will use the above mentioned inference rule. There is however no reasonable way to find the preconditions of the rule plausible except to add it to the assumptions of the context. This leaves us with three contexts, the outer one contains the reminder, the next contains the 'if' and in addition its assumptions list contains "John has money" and "John is in the shop". The innermost context contains just the conclusion.

4. It is now possible to close a context (before this it would have been a dead end since none of the contexts had effects and so could not have contributed to a reasonable interpretation of the surrounding context. The conclusion context is closed, causing the next context to have "John can buy some food" in its effects.

5. Now another context can be closed. First of all it's point must be determined, it has useful effects but it is does not have a 'conclusive' marker, so it will be assigned the point 'inference. Now, when *this* context is closed, the assumptions are *not* all plausible, there is no support for the assumption that John has money. Thus this context will be closed by the second 'conclusion' context closing rule, adding an inference rule to the world model of the outermost of the contexts to the effect that "if John has money he can buy food".

6. The next utterance is almost a reminder. In fact it is not the case that "John has his wallet" is true in the parent context (since the parent is the context of the reminder which started this text). Instead, since there is no was of inferring it and it has no special markers, it will have to be interpreted as an inference and it's plausibility supported by adding an assumption.

7. Next comes a conclusive utterance and so a conclusion context. When this is interpreted and closed we can also close the context created by "He has his wallet". This places John's having money into the first level context's world model (since The context just closed is not flagged as conclusive, it will be taken as an inference).

8. The final utterance is another conclusion, tying up the argument and giving the final discourse context an effect list containing only the fact that John can buy food. There is one thing in the assumptions list of the context we must now close, the assumption that John has his wallet. That fact is in the parent context and so is plausible and so this context can be closed as a simple conclusion, exporting "John can buy food" to the parent context's world model.

The major feature of this example is the illustration that the nesting of context is more than a way of ordering rule applications. Because only contexts with effects get closed and because those contexts were, except for the final one 'inside' the

contexts of other utterances, most of the inference done in interpreting this sub discourse is not 'visible' outside.

It is not clear that such a strictly imposed hierarchy of discourse context is a good model of human interpretation. Reichman [Reichman 1985] for instance, allows previously closed discourses to be reopened. However for the purpose of investigating how we can use an audience model in text planning it seems to err on the side of an over restrictive model rather than to create a model which gives the system no problems.

§7 Summary

Work in linguistics has provided information about the structure of connected texts and the relationship between speaker, text and audience.

Text structure provides a way for a discourse to reflect the goals of the speaker and for the audience to recover these intentions. Its interaction with global focus ties the information content of the discourse to the intentions of the speaker.

We have informally presented a simple model of text structure and interpretation which exhibits some interesting linguistic behaviour while remaining limited enough to be used by a practical text planning system as its audience model.

Chapter 3

Planning and Rational Action

§1 Introduction

This chapter discusses some of the models of rational action which have been developed within the field of Artificial Intelligence under the general heading of "planning" and examines their suitability as a basis of a model of linguistic action.

All planning models share a basic concern with formalising the relationships between means and ends in a way which makes them realisable as computer systems. They differ, of course, in many ways, but here we will be especially concerned with three properties —

- The type of model of the world which the planner maintains.
- The model of time and causality assumed by the planner.
- The way in which the system represents the relationship between itself and other agents.

Planning models have historically fallen into one of two classes, those which are presented as logical calculi with the implementation having the form of a more or less general inference system and those which are presented as a set of operations upon a structure which is interpreted as a plan together with rules for scheduling operations. The section §2 of this chapter presents the reasons for the choice of a structural rather than logical model of planning as a basis of the work described in later chapters.

§1.1 Why Planning

The planning paradigm was chosen as the basis of this work for a number of reasons —

- Since much work has been done within the planning framework, there exist a large body of established techniques from which to draw to solve problems which arise in creating a complete system.

- It is relatively well defined. This means it is fairly clear when some design choice departs from the basic planning model and allows such departures to be examined to determine if they are reasonably general extensions to the idea of planning or if they are motivated only by the needs of text planning, indicating perhaps an area where linguistic behaviour introduces problems not found in more general action.

- Certain phenomena of interest in text generation seem intuitively to parallel phenomena which are well studied in the planning literature. For instance avoiding redundancy in a text is very similar to avoiding unnecessary action in a plan and tracking possibly subtle implications of the planned text is apparently similar to the problem of planning for the possible side effects of a physical action.

- The work of Appelt, [Appelt 1982, Appelt 1988] and Hovy [Hovy 1985], discussed in more detail in chapter 4 has shown that planning provides a framework suitable for various text planning tasks.

The close fit between planning and language concepts can be seen by examining the Gricean maxims as presented in section §4.2 of chapter 2 from a planning perspective. The maxims of quantity say one should plan to say what will support one's goals and no more. The maxims of manner say that one should be careful to ensure that what is planned can be guaranteed have the desired effect when interpreted. These ideas are parallel to standard planning concepts of plan efficiency and plan correctness. The maxim of relation, and still more the reformulation by Sperber and Wilson, is a constraint on interaction with other agents which has direct parallels in planning for other agents to perform actions as we will see in chapter 5. In contrast, the maxims of quality relate not to general principles of action and communication, but to social norms and as such lie outside the scope of this work.

§1.2 Basic Planning Ideas

Although a large variety of models of action have been produced under the heading of planning, all of them share a broadly similar view of what constitutes rational action. An agent is modelled as having an explicit model of a state of the world which they wish to bring about and rational behaviour is seen as a process of comparing that goal state with the current state of the world and then constructing a plan of action which which can be acted upon and which will result in the goal state coming about.

The elements of the constructed plans are actions which are thought of as transformations of the state of the world. The choice of actions available to an agent is generally represented as a relatively small set of under-specified action types which are fleshed out into descriptions of specific actions by choices in the planning process and the context in which they are placed in the plan. For instance an agent might be assumed to know how to move an object from one place to another. The planner would decide which object should be moved based upon the goals of the agent and then the precise affects of the action would be determined by the state of the world at the time at which the action is to be performed, for instance where the object is currently, whether there is anything on top of it and so on.

We shall assume in this chapter that the reader is familiar with basic planning ideas. A good introduction to planning in general is [Georgeff 1987]. Logic based planning systems are described in chapter 12 of [Genesereth and Nilsson 1988].

§2 Structural vs. Logical Planning

Early work on planning, for instance [Green 1969], treated planning as a specialised form of deductive reasoning. It was rapidly discovered that reasoning about future events involves problems which are rather different from those encountered in more usual theorem proving domains. In these problems have been discussed under the general heading of 'the frame problem' and fall into two broad areas (following [Shoham 1988]) —

Qualification It is, in general, impossible to predict what is about to happen without taking into account an arbitrarily large amount of information about the current state of the world.

Extended Prediction The further into the future we plan the less reliable are the predictions as more and more contingencies might arise which would cause something unexpected to happen.

In a purely deductive planning system these problems must be tackled by providing the inference engine with extra information, usually called frame axioms. However the number and complexity of these extra axioms expands rapidly as the domain of interest gets more complex.

The identification of these problems led to the development of two complementary streams of planning research. Some researchers, such as Allen, Shoham and Nilsson, concentrated on making incremental changes to the deductive model of planning by changing the inference engine or creating new logics more suited to describing plans and planning mechanisms. Others, notably Sacerdoti in [Sacerdoti 1977], created systems where solutions to frame problems and issues of control were built into the algorithms and data structures of the system. We call this latter type of system 'structural' planners.

While the two approaches to planning are, to a great extent, simply different ways of viewing the same general model of rational action, it is clear that the choice of one or the other will influence essentially every other choice made in the design of the system. In the work described here it was decided to follow the structural path. A major factor in this decision was the complexity of the representation which forms the basis of the world model of the planner which would have been

difficult to implement efficiently in a general theorem proving system. Another factor was the desire to keep the distinction between the planning system and the text interpretation and inference system constructed within the planning system as clear as possible.

Work by Chapman [Chapman 1987] and Ginsberg and Smith [Ginsberg and Smith 1987] has provided a formal basis for standard models of structural planning, reducing further the difference between the two approaches.

§3 Models of the World, Goal Tracking and Conflict Detection

The models of the world used in planning systems are in general fairly simple. This is because the design of these representations has been dominated by the needs of the planning system rather than by the problems of representing the complexities of the real world. The planning process makes three strong demands on the world model it is to use —

- It must be possible to compare models of different states of the world and decide how they differ.
- It must be possible to determine if two models of the world are consistent, that is if they could both be models of the same state of the world.
- It must be possible to create a new model representing the result of executing a given action in the state represented by a given model.

Additionally, since a plan will contain many actions, and hence many states of the world, it is impractical to represent all states of the world by explicit models of what is or is not true. It must be possible to represent states of the world in a succinct way, for instance by giving differences from some base state.

§3.1 Simple Fact List Models

Much of the early work on planning, especially within the class of planning systems we call 'structural' planners, rather than logic based systems, concentrated

on issues of control of the planning process itself. Because of this the systems produced manipulated very simple world models, often simply a list of 'facts' which were held to be true at a particular point in time. Often such simple world models have been associated with structural planners such as Sacerdoti's NOAH [Sacerdoti 1977], but the choice of world model is orthogonal to the choice between logical and structural planning. For instance the logic based planner WARPLAN created by Warren [Warren 1974] used a simple world model because of its need to determine how goals are affected by intervening actions (goal regression).

This type of world model is simple to integrate and easy to compare with another model of the same type. However it is very limited in its ability to express complex relationships in the world and although it might be quite possible to represent interesting linguistic and inter-agent structures in such a format, the resulting representation would obscure rather than illuminate anything of interest in the model.

§3.2 Inferential Models

A quite different approach to world representation was taken by logical planning models such as that developed by Green [Green 1969] and Filkes & Nilsson [Fikes and Nilsson 1971]. With the full power of a logical inference engine available to them it was natural to apply that power to making the world model of the planner more sophisticated as well as to the planning process itself.

For instance, the original domain of application of the STRIPS planner was that of a robot moving from room to room and in this domain there were inference rules such as —

$$(\forall d\ \forall x \forall y)[\text{connects}(d, x, y) \Rightarrow \text{connects}(d, y, x)]$$

While this kind of system is attractive for its elegance, it does have major problems as a basis for text planning. The foremost problem is, paradoxically, that by integrating reasoning about the world with the planning process, such systems hide that reasoning from the planning process. The semantics of an inference engine ensure that there is no visible difference to the planning process between facts readily available to the system (or to its presumed audience) and facts which must be inferred. However it is important for a text planner to be able to reason about the reasoning needed to understand the text being planned.

A second problem is the difficulty of interpreting an inferential world model. For any reasonably complex inference system, questions such as 'if I add this fact, what will be inferable' and 'is this world model consistent with that one' are undecidable. This means that there is no ready way to decide which action might best achieve a goal or what conflicts exist in the plan without very extensive (potentially unbounded) reasoning.

§3.3 Hybrid Models

The complimentary limitations of simple fact-lists and full inference systems as world models for planning naturally inspired researchers to develop systems which occupy the middle ground between the extremes. There are two main types of hybrid systems, those which provide an inferencing world model to the planning system, but separate it out so that the planner can more readily take the inference into account, and those which integrate a limited form of inference into the planning process in an attempt to gain some of the flexibility of an inferencing world model without losing the efficiency of a directly interpretable representation.

KAMP, a system developed by Appelt [Appelt 1982] for use as a text planner took the first approach. The world model which KAMP uses to construct its actions is active; that is an inference engine independent of the planning process is invoked each time a world model must be queried. This mechanism has advantages and disadvantages from the point of view of the text planning process as we will discuss in section §3.3 of chapter 4. It also causes some problems for the planning process. Because KAMP's planning engine can not predict what effects making some fact true in the plan may have, it can not in general determine from the definition of an action whether it will be useful in a particular situation.

Actions which have the effect of changing the beliefs of other agents will have large numbers of effects which may change depending on the situation in which the action is performed. Often it will be difficult to plan actions because of these "side effects" of its linguistic actions. Appelt solved this problem by giving KAMP, in addition to the logical definition of the results of an action, a summary of the action which gives the usual results of the action in a more direct way. This leaves problems when the action is planned under unusual circumstances, when it might have unexpected results, and so Appelt added to the classic non-linear planning algorithm, as exemplified by NOAH, an extra critic which calculates the actual effects of the actions in the plan and checks that these are close enough to what

was expected not to cause problems[1]. For instance, the following is one of the post-conditions of the 'Request' action used by KAMP —

$$\begin{aligned}\forall A, B, P, w_1, w_2 R(:Do(A, :Request(B, P)), w_1, w_2) \rightarrow \\ [\forall w_3 K(Kernel(A, B), w_2, w_3) \rightarrow \\ \exists w_4 K(Kernel(A, B), w_1, w_4) \wedge \\ R(:Do(A, :Request(B, P)).w_4, w_3)]\end{aligned}$$

Formulae of this complexity put a large strain on the deductive system and, since conditions must be proved regularly during the planning process, KAMP must engage in a great deal of expensive reasoning for even simple plans.

Wilkins' SIPE system described in [Wilkins 1988] takes the second approach to creating a hybrid system. The action definitions given to SIPE have their pre-conditions and results described in a language which allows simple quantification and context sensitivity, but which is carefully limited in power so as to allow the planning process to effectively determine the expected results of an action. In this way SIPE escapes the problems of undecidability which KAMP suffers. However, the limited power of the 'logic' available for describing actions does mean that it seems unlikely that an interesting audience model for text planning could be described in that logic.

§3.4 World Models for Text Planning

It can be seen from the preceding discussion that the choice of world model for a planning system is critical. In section §6 of chapter 2 we presented a simple model of text interpretation which we wish to use as the audience model of a text planning system and naturally we wish this to be easily representable in the world model of our planner. The complexity of this representation together with our need to track the details of the interpretation process argues against having an inferential world model.

Fortunately, there is a way in which we can gain some of the features of both simple and inferential world models given that we will have a system capable of planning for the actions of other agents. If the inferential behaviour of the audience is

---

[1]That is, all effects which where required as preconditions for later actions are achieved and no extra effects cause destructive interactions in the plan

treated in the same way as any other behaviour, the planning system will be able to plan for audience inferences in the same way as it might plan for the audience leaving the room or picking up an object.

§4 Time and Causality

The models of time which have been used in planning systems can be broadly divided into two classes, those based on the concept of a point of time and those based on the concept of a period of time. These two types of model are duals of one another since a time span can be represented by its start and end points and a point as the meeting place of two spans. However the two views give rather different perspectives on the planning problem.

§4.1 Time Point Models

Models of time which use points of time and their relationships have been used in many planning systems. Perhaps the most famous example is NOAH which represents its plan as a network of time points which correspond to actions and goals (in effect, under-specified actions). Other non-linear planners such as NONLIN [Tate 1976] and SIPE [Wilkins 1988] followed NOAH in this respect.

Representing time by specifying points and the relationships between them has the advantage of making the temporal relationships such as before and after simple to determine. However, planning cannot in general be done entirely in terms of time points since it is often necessary for the planner to be able to specify that some constraint holds for some period of time. For instance when an action is planned to provide a precondition to some other action, the system must ensure that the precondition will remain true between the times of the two actions. For simple linear plans it is possible to ensure that such constraints are maintained by careful control of the planning algorithm, for instance WARPLAN [Warren 1974] used goal regression to ensure that when actions were moved within the plan any interference from other actions would be avoided. However, this kind of care becomes unreasonably complex as non-linear plans and more complex world models are introduced and so many later systems incorporated extra mechanisms (for instance the "resolve conflicts" critic in NOAH) or representations (for instance NONLIN's 'goal structure') to keep track of these constraints.

§4.2 Time Span Models

Planning systems which are built upon the basis of a time-span model of time can be seen as making the implicit assumption that the important aspect of a plan is the representation of how the planner expects the world to change over time. Any actions which are performed by the agents in the domain supply reasons for the changes in the state of the world described in the plan and hence evidence for the plan's correctness.

Most of the work on such systems has been done within the framework of logic based planning. There have been many proposed span-based temporal logics, for instance [Allen 1984a], [Dean and McDermott 1987]. Planning systems based upon such representations have had to deal with the fact that planning with ranges, like non-linear planning, is non-monotonic. That is to say, as planning progresses, later decisions can make earlier ones incorrect. Dean and McDermott handle this non-monotonicity by building their logic upon a truth maintenance system [Doyle 1979] and having the planner manipulate the resulting non-monotonic representation, while Steel, [Steel and Leung 1989], integrates a assumption based truth maintenance system [DeKleer 1986] into the planning algorithm. Both systems gain something in clarity from having the non-monotonicity 'out in the open' in the plan representation and its manipulations rather than having to deal with separate critics or a separate constraint maintenance system.

Another approach to planning with time spans is that of [Reichgelt and Shadbolt 1989] who adapted the concept of reasoning as theory extension developed by [Poole 1988] to planning. The central idea is that planning can be seen as developing a theory of what the future course events could look like if it is to contain the states of affairs which are the goals of the system.

§4.3 Choosing a Model of Time for Text Planning

The nature of text planning is that it involves relatively small plans (a complex linguistic task might be performed by making only two or three utterances and planning for the audience to make half a dozen or so interpretive actions) involving complex changes in the planner's world model and complex constraints. For this reason it seems natural to choose a model of time and plans which emphasises the relationship between *states* rather than between actions. The model which will be described in chapter 5 is a time period based representation using a single

primitive to represent action preconditions and effects and also constraints on the plan.

§5 Other Agents and Interaction

In order to make a reasonable attempt at text planning, a planning system must be able to deal with the presence of at least one other agent — the intended audience. Early work on planning concentrated on single agents operating in simple passive environments. Later work which allows for the presence of more than one agent can be divided into two types —

- Systems which plan actions for a number of agents all of which are assumed to be under the control of the planner.

- Systems which plan actions for one agent but take into account the expected actions of other agents outside the control of the planner.

A system which is to plan linguistic actions will be of the second type. Although it is possible to think of a discourse as being a cooperative game between the speaker and audience, such a third person model will be primarily descriptive, at heart an advanced form of discourse grammar and provide few if any opportunities to explore the relationship between language and general rational action.

The remainder of this section discusses some other issues which are raised by even the simplest attempt at planning in a multi-agent environment. In general the focus of this work has not been upon planning issues for their own sake and so we have tried, where possible, to favour simple solutions which provide the functionality actually required.

§5.1 Representing Other Agents' Planning Behaviour

If a planning system is to treat other agents as rational actors like itself then it must be able to some extent at least, to represent their plans and planning behaviour. In general this might mean that at each point in its plan, the system would have to maintain a model of what it believes the expected future plans of every agent in the domain, possibly including itself, will be at that time. The DePlan system

[Hopkins 1989], for example uses such information to delegate planning tasks to other agents. While such generality might be necessary for very complex situations where agents' beliefs and desires change rapidly, we assume here that for simple text planning tasks a much simpler solution is possible.

We choose to include other agents' expected actions in the systems plan on a par with its own actions. This means that the system will be able to plan for a particular action to be performed at a given time by some agent without having to deal with exactly how that agent decides to perform that action. Of course the system must have some model of how other agents' plan so as to be able to predict which actions they will perform and to be able to influence them to perform the actions it needs. Our solution is to define an extremely abstract model of how a rational agent might be expected to behave based on ideas from linguistics discussed in section §4.2 of chapter 2. From these ideas we extract a concept of an action being 'motivated' for some agent at some point in time. 'Motivation' is described in detail in section §5.4.5 of chapter 5.

§5.2 Assigning Responsibility and Scheduling

A second source of complex problems in multi-agent planning systems is the necessity of assigning responsibility for achieving various goals to different agents and scheduling their actions so as to avoid conflicts and ensure that each agent is capable of performing the actions scheduled for it when requested.

A great deal of work on these problems has been done under the general heading of 'Distributed Artificial Intelligence'. For instance [Cammarata et al. 1983] discuss different task allocation schemes which might be used in the domain of air traffic control. Fortunately the problem of text planning and interaction with an audience is such that we can avoid many of the more complex issues.

Since most of the actions which we will be planning are inferences and other interpretive actions by the audience, the problem of deciding who should perform them does not arise — the speaker obviously can not infer things for the audience. Similarly precise scheduling of the audience's actions is not necessary so long as

we can assume that all of the interpretation for one utterance will have been completed before the next utterance occurs[2].

A related problem is that of which agent is responsible for supplying the preconditions of a given action. In this case the problem does have interesting linguistic consequences since the more responsibility is placed on the speaker, the more wordy the planned texts will be. The rules we adopt are described in section §5.4.3 of chapter 5 and are designed to make the speaker responsible for making sure that preconditions directly related to their goals are supplied while leaving the audience to deal with goals which arise as a side effect of the speaker's plans. Different responsibility allocation rules will be required by different types of linguistic behaviour, but the simple ones which we adopt seem to suffice for the types of text we are interested in.

§6 Summary

In this chapter we have examined the various types of system which have been proposed by workers in Artificial Intelligence as models of rational action and planning and have made some choices based upon the requirements of the text planning task. Briefly, those choices were —

- Structural rather than logic based planning.
- A complex but passive world model with the active nature of the audience of a text being modelled by the planning process itself rather than by a separate inference engine.
- A time span based model of time rather than one based upon points of time.
- Fairly simple solutions to the problems of scheduling actions and assigning responsibility based upon the properties of the text planning task.

The details of the planning model we constructed based upon these decisions are presented in chapter 5.

---

[2]Obviously this might not be true in situations where the audience is under time pressure, but for normal interactions it seems to be a reasonable simplifying assumption.

Chapter 4

Language Generation and Text Planning

§1 Introduction

Most existing work on the problem of language generation has concentrated on the problems of deciding the surface form of a text whose content, and sometimes organisation, is assumed to have been decided by some outside agency. In this chapter we start by examining these 'tactical' generating systems to determine what decisions a text planning system must make and what information it should provide.

Then we examine some of the existing work on text planning and try to determine the problems with these systems and suggest some solutions to be taken up in later chapters.

§2 Tactical Generation

The very simplest systems providing natural language output do so by having some repertoire of pre-stored texts, choosing one and outputting it. The 'help' facility of many computer systems works in this way (For instance the UNIX[1] operating

[1] UNIX is a trademark of AT&T Bell Laboratories

system's on-line manual or the help command of the POPLOG[2] programming environment). In such a simple case the input to the 'generator' can be thought of as simply one of a fixed number of tokens each corresponding to a 'canned' text.

Rather more sophisticated output can be produced by a system which stores not fixed texts, but templates giving the outline of texts but with gaps which can be filled differently depending on the instructions which the generating system receives. One of the commonest application for this type of generation is in the production of error messages in computer programs. Typically such messages are short and simple but not totally fixed, since they must describe the context in which the error occurred. The input to this type of system would consist of a token indicating the the message to be produced together with a description of the items to be inserted into the gaps in the template. An example of such a description might be —

$$\textit{undefined-name-message}(\textit{line-number} = 143, \textit{symbol-name} = \textit{Foo})$$

Given such a description the system might produce a message like

`Undefined symbol 'Foo' at line 143`

where the underlined sections are the result of inserting the parameters into the template.

§2.1 Grammar Based Tactical Generators

The text produced by the kind of template system described above is, of course, very limited. However, the basic concept can be extended to produce a much more flexible tactical generation system.

- The templates could encode, in addition to fixed text and parameters, restrictions on the parameters (for instance to enforce number agreement).
- The templates could be made more flexible by making more of the constituents parameters, rather than fixed text. The ultimate form of this is to replace the fixed text entirely and turn the templates into structures resembling phrase structure rules.

---

[2]POPLOG is a trademark of the University of Sussex

- The handling of parameters could be improved by using the techniques we are applying to the text as a whole (templates and restrictions) to produce natural language realisations of the parameters.

- The connection between the structure of the input and the template could be loosened by storing with each template instructions about how its parameters should be extracted from the input.

- Similarly the explicit encoding of the template name in the input could be removed and the template could then contain information about what types of input it is useful for.

Systems which have this structure have been produced by a number of people. Jacobs' PHRED system [Jacobs 1985] manipulated structures of the form described above, using unification to recognise useful templates, extract parameters from the input and enforce restrictions. McDonald's MUMBLE system [McDonald 1981] has much the same structure, though rather than creating linguistic output directly it builds an intermediate structure which is then further processed to produce the actual output. The tactical components of McKeown's TEXT [McKeown 1982] and Appelt's TELEGRAM [Appelt 1983] both use Functional Unification Grammar [Kay 1986] to represent this kind of pattern-structure relation.

In this type of system the input to the tactical generation process is a description of the information to be conveyed in some internal representation possibly together with some information about how this information fits into the larger structure of the text being produced. For instance the tactical component of TEXT uses a functional unification grammar and the input consists of a functional structure representing a case frame and some focus information.

§2.2 Systemic Grammar based Systems

A rather different approach to generation has been been followed by a number of systems based on ideas from Systemic Grammar[3]. Systemic Grammar places emphasis on the choices which a language gives to a speaker rather than on the structure of utterances, these choices being described using "system networks"

---

[3]A good overview of Systemic Grammar can be found in [Winograd 1983]

which indicate relationships between these choices. The relationship between these choices and the surface form of an utterance is described by a separate set of "realisation" rules which place restrictions on the relative positions of parts of the utterance.

As might be expected, generating systems which have been developed using ideas from Systemic Grammar have tended to place far more emphasis on the problem solving nature of language use (trying to find a set of choices which will produce a text with some required characteristics) and less on the problems of constructing complex linguistic structures.

Davey's PROTEUS system [Davey 1978] used a systemic grammar to guide its choices in producing a description of a game of noughts and crosses. The Penman system [Mann 1983] is built around a large systemic grammar and a collection of choice procedures which guide the search for a path through the system networks. Patten [Patten 1986] explored the relationship between Systemic Grammar and traditional AI problem solving techniques.

Systemic Grammar's advantage as a basis for generation lies in its ability to describe how orthogonal but interacting sets of choices combine to select the form of the final text. In tactical generation this type of interaction occurs most strikingly when the constraints on the focus of a clause combine with its information structure to cause the selection of a marked structure (for instance a passive or topicalised sentence).

Although more concerned with interpersonal and informational issues than the systems described in the previous section, the generators based on systemic grammar can still be seen as taking as their input information at a similar level of abstraction.

§2.3 Conclusions

From this very brief review of work in generation of surface natural language utterances we can see that although the systems vary greatly in structure and linguistic formalism, there is broad agreement about the level of abstraction at which the task of language *generation* changes into that of language *planning.* This level, that of non-linguistically based semantics and pragmatic markers will be treated, for the remainder of this thesis, as the lowest level in which we are interested.

§3 Text Planning

Less work has been done on text planning than on surface generation. Power's work on conversational games between two agents in a simple world [Power 1974] explored the relationships between an agent's goals and abilities, their beliefs about another agent and their conversational behaviour. However the conversational moves which the agents had available were very limited and lacked complex semantics.

Davey's PROTEUS system [Davey 1978] was an early attempt to produce a system which generated texts to achieve an objective (to explain a game). It produced reasonably fluent text structured to reflect the flow of the game.

OSCAR [Cohen 1978] was developed by Cohen to plan sequences of speech acts to achieve some goal. However it did not attempt to determine the surface form of the speech act in any way, nor did it have any concept of text structure.

Mann & Moore developed a system, KDS [Mann and Moore 1981], which used heuristic, hill-climbing methods to decide on a structure for a paragraph sized text (instructions to be followed in case of a fire). However many of the heuristics which the system used were specific to this kind of text and do not generalise well, for instance —

> **5.** Certain constructions get bonuses of 20: the if-then-else construct and the when-X-determine-Y. *(page28)*

All of these systems are very specialised. Some of the more recent work on text planning has attempted to develop more general theories of text construction. In the following subsections we will discuss some of this work in more detail. Three models will be examined —

- McKeown's TEXT system.
- Rhetorical Structure Theory and Hovy's work on implementing this model using a hierarchical planner.
- Appelt's KAMP system.

These systems might be seen as lying on a spectrum from TEXT's rather rigid schema based approach to the very general planning and reasoning system of KAMP.

§3.1 TEXT

McKeown's TEXT system was designed to answer queries about the structure of a database. It planned texts by selecting a schema, based on the type of question, and using that schema together with constraints on the movement of focus within a text to guide its choices.

The schemas were the result of an analysis of human produced texts. Each schema was in the form of an ATN were the arcs where labelled with "Rhetorical Predicates" such as 'Evidence' or 'Analogy'. Associated with each of these predicates was one or more functions which were specialists knowing what information from the knowledge base should be output to perform that action. As the schema were traversed each arc crossed caused the functions associated with the predicate labelling that arc to be called. These functions collected information from a pool of "relevant" facts; this selected information formed the input to the tactical generator. When the schema left open a choice (either because more than one arc in the schema was a possible next move or because a predicate has more than one function associated with it) then focus information was used to choose a course.

The focus model used by TEXT was based on the work of Grosz and Sidner. When starting a schema TEXT selected a collection of "relevant knowledge" (using heuristics based on connectivity in the knowledge base) which took the place of a "focus space" in Grosz's focus theory. Only the information in that "space" was available to the system to construct the utterances required to traverse the schema. In addition to this TEXT maintained a local focus list, patterned after the work of Sidner, which further restricted the search for a path through the schema.

TEXT had a number of limitations. Most strikingly it made no attempt to model its audience so it would always, given the same query, give the same response. Similarly it did not maintain a model of the past discourse.

Another limitation came from the schemata themselves. Schemata were large, hand crafted, structures specialised for one task only. Their size combined with the fact that the system had no information about *why* it applies a given schema or rhetorical predicate makes it unlikely that the system could easily be generalised

to cope with situations for which it was not designed or for domains where the problems which it would be set were more varied.

Despite these limitations, TEXT was capable of producing reasonably fluid and relevant text in a domain where the amount of information available to the generator is large and the generator's main problem is selecting and arranging that information. This fluency comes from two sources

1. The schemata, being based on human produced text, embodied a large amount knowledge of how short descriptive texts should be constructed for an unknown audience.

2. The focus model guided the planning of the text, producing a coherent structure to the presentation of the information

The first of these sources of fluency is not available to a system which is designed to be more flexible, since it would be impractical to build schemata covering every possible circumstance which a system might be placed in. For such a system we must make explicit the knowledge of how a text should be structured and why it should be structured that way.

A knowledge of the way focus moves in a text, on the other hand will be useful in any situation. The model of interpretation described in section §6 of chapter 2 takes the notion of global focus to be central. Local focus has not been tackled in the work presented here because it is intimately connected with issues such as the structure of referring expressions which we do not tackle. TEXT presents a strong argument that this is an area where future work would be fruitful.

§3.2 Rhetorical Structure Theory

Rhetorical Structure Theory (RST) was described in section §3.2 of chapter 2. Hovy [Hovy 1988] has worked on applying RST to form the text structure knowledge of a text planning system.

He recast the constraints and effects as statements about the beliefs of the speaker and audience expressed in a logical formalism developed by Cohen and Levesque [Cohen and Levesque 1985]. Figure 4–1 shows the constraints which he places on the "purpose" schema used to represent text fragments which describe the purpose of an action.

Nucleus Constraints:
1. (BMB *S H(ACTION ?act-1)*)
2. (BMB *S H(ACTOR ?act-1 ?agt-1)*)

Satellite Constraints:
1. (BMB *S H(STATE ?state-1)*)
2. (BMB *S H(GOAL ?agt-1 ?state-1)*)
3. (BMB *S H(RESULT ?act-1 ?act-1)*)
4. (BMB *S H(OBJ ?act-2 ?state-1)*)

Intended Effects:
1. (BMB *S H(BEL ?agt-1 (RESULT ?act-1 ?state-1))*)
2. (BMB *S H(PURPOSE ?act-1 ?state-1)*)

BMB = "Believe to be mutually believed"

BEL = "Believes"

Figure 4–1: The "Purpose" relation

Hovy's text planning system uses these formalised relations and the RST schema as planning operators in a hierarchical planning process similar to that of NOAH to construct a structure for the text. The final planned text is passed to the Penman tactical generator for output.

RST structures form a better representation for the text structure knowledge of a generation system than McKeown's schemas since they are smaller, and so can be used more flexibly, and because they explicitly encode the *purpose* of the structures they describe. A system which does not have such knowledge can not possibly cope with unexpected situations.

The audience modelling performed by Hovy's system is very basic. Although the formalism he uses for constraints and effects allows him to describe the audiences beliefs and the direct effects on those beliefs of the various text structures, there is no model of how the audience will react to the changes in their beliefs. The user model is totally passive and though work is in progress on how the options for text construction provided RST can be influenced by focus [Hovy and McCoy 1989], this is being done from the point of view of the speaker and not as a way of tracking the audience's attention.

§3.3 KAMP

Appelt's KAMP system performs text planning using a general purpose non-linear planner combined with a first order theorem proving system. We discussed some

of the features of KAMP from a planning perspective in section §3.3 of chapter 3. In this section we describe how the mechanisms used by KAMP influence its performance as a text planner.

In order to be able to reason about the beliefs of multiple agents, KAMP uses a representation for multiple beliefs based on that developed by Moore [Moore 1981]. In this representation beliefs are not represented explicitly, rather statements are made about relationships between sets of possible worlds which represent the range of options for how the world might be allowed by an agents beliefs.

This representation has a number of advantages. It allows for the description of disjunctive beliefs and of mutual knowledge in a well defined way, it can be used to represent temporal as well as doxastic information and has a close fit with planning ideas (if we interpret states of the world in a plan as possible worlds). However there are disadvantages.

The representation of beliefs as sets of possible worlds makes it impossible to model the reasoning *process* of the audience. Although the audience model which KAMP supplies is active, in that if KAMP knows that the audience has been given some information it will know that they know other things by inference, it does not have any model of *how* the audience performs the inference leading to that extra information., which could be important if the system wishes to block some of that inference.

A related problem with this representation is, as mentioned in chapter 2, that of "logical omniscience". All agents in domains modelled by KAMP are assumed to believe all valid logical consequences of their beliefs; that is, their beliefs are closed under logical deduction. This is obviously not true of human readers and causes a large mismatch between the model of language use presented by KAMP and actual human language use.

KAMP does not have any notion of text structure since it was designed to plan single complex speech acts satisfying multiple goals. However, for longer texts, knowledge of text structure is essential both to making the text seem coherent to a human audience and to limit the search space of possible texts.

§4 Conclusion

The problem of text generation as a whole may be usefully divided into sub-problems. A major division being between the problems involved in selecting and arranging the contents of a discourse and the problems of deciding the surface realisation of the structural choices. A natural representation of individual utterances for the higher level decision making processes is a case based description of the "content" of an utterance together with information on focus.

A model of discourse planning must be able to integrate many different sources of information. Important parts of the discourse planning process are —

- Structuring the discourse to reflect both the speaker's intentions and the conventions of the type of discourse being planned.
- Planning the movement of the local focus of attention within the discourse to produce a coherent text.
- Tracking and taking advantage of the audiences behaviour as they interpret the discourse.

Appelt's KAMP system proved that AI planning techniques can be used to perform the latter two processes; however the representation which KAMP uses for beliefs is such that it models its audience at a level of abstraction which prevents it from guiding the details of the audience's interpretation.

Chapter 5

A Model for Planning and Communication

§1 Introduction

In this chapter we describe the problems posed by a language generation system for the representation and processing model upon which it is built, and sketch, in outline, some possible solutions. Our basic strategy, as discussed in chapter 4 will be to take mechanisms from work on planning systems and extend them when necessary to cope with the specific problems which arise in planning communicative actions. This chapter is intended to provide the motivation for the description of the Riple system in chapter 6 and the more rigorous presentations in appendices A, B and C.

The problems are of several types —

Expressiveness In a domain which must contain multiple interacting agents, possibly with different, or even directly contradictory, beliefs about the state of the world and each others beliefs, some way must be found of representing multiple, divergent, world models in a way which makes the represented information quickly available to the reasoning of the system. Also, since we wish to be able to implement interpretation models such as the 'points' model, described in section §6 of chapter 2, the model must provide the necessary expressive power.

Generality In order to constrain the problem which the system tackles to a manageable size, assumptions must be made about the domain in which it works and the type of mechanisms which it must try to describe. However, if the system is to be of more than passing interest it must not make assumptions which will not generalise to larger and/or more complex domains and linguistic phenomena which it does not try to cover.

Efficiency Although not a high priority, efficiency is important in so far as the implementation of the proposed model should not demand any obviously intractable computations to be performed.

Another criterion guiding the development of the mechanisms described here has been the belief that *in the long term* a goal of the design of planning systems should be to produce systems capable of reasoning about their planning behaviour. [Wilensky 1983] describes a system able to do this to a certain extent. Although we have not attempted to produce such a self describing system here, some care has been taken to ensure that the mechanisms we propose clash with this goal as little as possible.

§2 Basic Assumptions

There are certain basic assumptions about the type of domain about which a language-using system reasons and communicates which, although not necessarily trivial or even correct, are taken for granted in the work described here —

Discreteness The universe of discourse within which the system works is assumed to be made up of discrete 'entities' having well defined properties and standing in simple relationships with one another. Similarly the passing of time is assumed to be marked by the occurrence of discrete events causing well defined changes in the state of the world.

No Parallelism Events must occur in a well defined order — there is no provision for overlapping events. This means, in effect, that all events are assumed to be instantaneous.

Modularity The universe is assumed to be modular. That is, it is assumed that it makes sense to describe and reason about relatively small subsets of the universe while ignoring what is happening to other parts.

Agent Similarity All the agents who exist in the universe are assumed to be of a similar kind. In particular, we assume that the system can know the actions possible for other agents, the rules which govern other agents' adoption of new goals and the order in which another agent will consider different possible actions in the attempt to find one which will have a desired effect.

Subjectivity The system is assumed to be taking the part of just one agent in the universe. This agent is the only one about whom the system has direct knowledge and is the only one whose actions it can select.

These assumptions are made in order to remove some of the very difficult problems which exist in reasoning about dynamic environments containing multiple agents. These assumptions are the most obvious way in which the system described here is not easily generalisable. Removing any of these assumptions would lead to major increases in the complexity of the problem which the system faces.

§3 Elementary Requirements

The most elementary task which any reasoning system faces is to represent and manipulate the entities which form its universe and their properties and relationships at different times. In this section we describe some of the basic properties which a representation which is to be used by such a system must possess.

§3.1 Entities

The basic assumptions described in section §2 constrain the entities with which the system must deal to be simple, discrete 'things' which can be completely described by their properties and the relations in which they stand to other entities. In the case of a language-using system many of these entities will be more abstract than the standard 'blocks world' type of object with which many AI reasoning systems are concerned; concepts such as 'utterances' and different agents' concepts of some object form a very important part of the domain. However, for most purposes these abstract objects need be treated no differently from more concrete ones, such as the agents taking part in a discourse.

Typical entities which the system might need to represent are —

7) John.

8) John's concept of Mary's car.

9) The utterance Mary has just made.

10) The English past tense.

11) The verb "shine".

Since the representations of 'entities' which the system must manipulate are atomic, in the sense of having no internal structure, we need only ensure that the chosen representation produces a distinguishable token for each such entity. In later examples we will use simple tokens such as —

7) **john**

8) car2

9) utterance7

10) past_tense

11) vshine

to stand in for whatever representation is chosen for these concepts.

Some of the objects which a language user must manipulate are more complex than the sort described here. There will usually be a need to talk about and manipulate complex structures such as discourse contexts and (partial) descriptions. See the discussion in section §4.2 about modalities for a description of how these, more complex, concepts can be represented.

§3.2 Properties and Relations (Terms)

Properties and relations, like entities, need only be very simple given the basic assumptions which we make. However, unlike entities, relations and properties are not atomic; there are meaningful questions we can ask about a property or relation beyond determining its identity —

What type? A relation is an example of a type of relationship, for instance "larger" or "on top of". Similarly, there are types of property "blue".

Which entities? A relation exists between some number of entities. A property must be a property of some entity.

The similarity between relations and properties in terms of the questions which can be asked about them leads us to classify them together as specialisations of a more general concept which we call 'terms'; properties are simply terms which have only one entity as a component. A representations for terms needs to allow us to answer these questions quickly and simply. The most straightforward way to provide for such queries is to use the type of representation for terms which has long been standard in logic, consisting of a functor, which is just an atomic object representing the type of the term and a sequence of entities.

For instance the terms,

12) John is on top of Mary's car.

13) Mary's car is Blue.

14) This utterance is in the past tense.

might be represented by structures which we could write as

12) **on (john, car2)**

13) **blue (car2)**

14) **tense (utterance7, past_tense)**

Here the functor is written outside the brackets and the entities within, in order. Since each functor is used in a well defined class of terms, such as utterance — tense relationships for the 'Tense' functor, properties of the class, such as the number of elements which participate in a relation (its 'arity'), can be associated with the functor.

§3.3 Representing States of the World

A system which is to perform actions to achieve its goals must be able to represent and reason about various states of the world which might come about as a result of its actions. The system must be able to perform a number of manipulations on the representation —

Term evaluation What is the status of a term in a given state, for instance to determine if a certain linguistic action is a good choice.

State comparison What do states have in common and in what ways they are different. This would be necessary, for instance, when trying to determine what needs to be changed to reach the system's goals.

State construction Postulating states or classes of states and reasoning about their properties in order to consider states of affairs which have not yet occurred.

State constraint Finding those states which satisfy some set of constraints and reasoning about their properties. Such reasoning would be needed to decide on the likely outcome of an event (linguistic or otherwise). In such a case the state of affairs before the event and the properties of the event itself will combine to constrain the resulting state of affairs.

A way to represent states which allows for these kinds of manipulation is to build descriptions of them by assigning values to terms. A class of states is represented as a collection of terms, each of which is explicitly given a value, such as true or false or something more complex, as described in section §4.2. A single state can then be viewed as a description which assigns a value to all possible terms.[1] However since any system must always work with only limited information, it never needs to manipulate descriptions of single states, rather there will always be uncertainty about the precise state of the world at a particular time and so the system must manipulate classes of states, and hence finite structures. From here onwards "state" will be used to mean "class of states" in this sense.

For example take the class of states described by the following English paragraph —

> John is sitting in his car, which is blue, and smoking. Mary is standing outside the car and is saying "hello".

This can be viewed as asserting a number of things about the world (ignoring for the moment the problems of identifying John and Mary and determining which car is John's).

- John is sitting.
- John is in the car.
- The car is blue.
- John is smoking.

---

[1]This is similar to the states of [Levesque 1984] or to the situation types of [Barwise and Perry 1983].

```
sitting (john) = TRUE
in (john, car1) = TRUE
blue (car1) = TRUE
smoking (john) = TRUE
sitting (mary) = FALSE
in (mary, car1) = FALSE
speaking (mary, word_hello) = TRUE
```

Figure 5–1: A State Description

```
  sitting (john)
  in (john, car1)
  blue (car1)
  smoking (john)
¬ sitting (mary)
¬ in (mary, car1)
  speaking (mary, word_hello)
```

Figure 5–2: The State Description Using '¬'

- Mary is not sitting.
- Mary is not in the car.
- Mary is saying the word "hello".

This could be represented by defining functors for the different types of term, 'is sitting', 'is blue' and so on, and constructing a description assigning values to terms as shown in figure 5–1. Since each term in such a description may have only one value, in this case either TRUE or FALSE, we might write such a description in a more condensed form by simply writing down those terms which are assigned a value and preceding those which have value 'FALSE' by a logical negation symbol '¬'. This is done for the same description in figure 5–2.

Using a representation of this kind the manipulations described at the start of this subsection can be done quite simply.

Term evaluation The status of a term can be directly read off from the description. Notice that although the notation given above

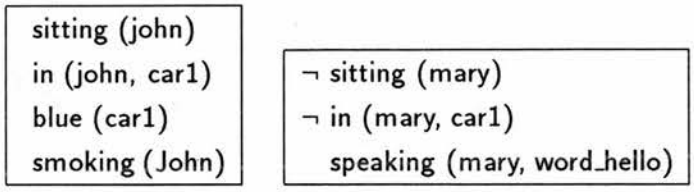


Figure 5–3: Two State Descriptions

resembles a logic, there is no notion of inference or reasoning involved in its interpretation. Only those terms which are explicitly given values in a description will have values. See the discussion of equality in section §4.3.

State comparison Comparing states is equally simple, just list those terms which exist in only one description and those with a different value in each.

State construction New state descriptions can be built quite quickly and easily by just giving the status of each term which is known.

State constraint Given two state descriptions such as those shown in figure 5–3 we can build a description of those states where both descriptions hold by simply concatenating the descriptions to get the one given earlier (The order in which the value assignments are written down does not make any difference to the meaning of the description).

§4 Complex Worlds, Modalities and Multiple Belief Sets

The simple interpretation model described in section §6 of chapter 2 maintains a much more complex notion of the current state of the discourse than the simple list of terms and values we have presented so far. In this section we discuss some of these extra needs and extend our representation to deal with them.

§4.1 Entity Creation and Destruction

One type of event which cannot occur in the simple domains often chosen for planning systems is the creation of new entities. In the blocks world the domain is defined with a given set of objects and these are all there will ever be. However in a domain involving communicative actions entities can come into being at any time. Some examples of entity creation are

Making an utterance An utterance is an event in time. In a very real sense it does not exist until after it is performed.

Indefinite reference Reference using an indefinite determiner or an existential quantifier must usually be interpreted as referring to some entity which cannot be identified with any entity which is known to exist, although it may be later determined that the new entity is simply another concept for a entity already known.

In order to cope with this kind of problem we need a representation which describes not just what is known about a particular state but also what entities are known to exist. There are a number of ways of doing this — for instance we could associate a set of entities with each description or there could be a functor 'exists' which is used to indicate existence. We need not commit ourselves to a particular representation at this stage, we can just devise a schematic description for these descriptions and their information about existence. For simplicity we choose to list the entities which exist in a state in a box at the top of the state description.

As an example, take the state description from section §3.3

> John is sitting in his car, which is blue, and smoking. Mary is standing outside the car and is saying "hello".

If we assume that the only entities which are known to exist are those explicitly mentioned then we might represent this knowledge as shown in figure 5–4.

If the system was to interpret and accept the utterance

15) Mary sees a policeman

This might cause it to believe that the state of the world was as described by the description in figure 5–5. Here, not only has a new term been given a value, but a completely new entity has been included in the description.

john mary car1 word_hello
sitting (john)
in (john, car1)
blue (car1)
smoking (john)
¬ sitting (mary)
¬ in (mary, car1)
speaking (mary, word_hello)

Figure 5–4: A State Description With Entities

john mary car1 word_hello policeman1
sitting (john)
in (john, car1)
blue (car1)
smoking (john)
¬ sitting (mary)
¬ in (mary, car1)
speaking (mary, word_hello)
sees (mary, policeman1)
police_officer(policeman1)

Figure 5–5: After Interpreting "Mary sees a policeman"

§4.2 Modalities

The domain in which a language user must work is rather more structured than can be represented by the kind of simple description provided for above. A range of phenomena will necessitate the classification of terms in a more fine grained way than can be achieved by simply assigning them boolean values.

Discourse Contexts It is important that a distinction be made between those things which the system believes to be true and those which it is simply manipulating in order to interpret a ongoing discourse. A discourse will introduce entities, assert terms about them and invite inferences. All of this must be done without necessarily affecting the actual beliefs of the participants in the discourse since, for instance, a discourse might

be purely fictional or it might contain an argument in the form of a 'reductio ad absurdum' and so tentatively assert things which are known to be false. For these reasons, the system must be able to describe, in addition to the current state of the world, the state as presented in the discourse in which it is engaged. We call such a description a 'discourse context'.

Assumptions The 'points' model of chapter 2 allows the audience to make arbitrary assumptions in order to find an interpretation for a text. To do this it must be possible to keep track of these assumptions.

The Beliefs of Others When it becomes necessary to record the beliefs of other agents (see section §4.3) more structured world descriptions will be needed.

Rules of Inference The definition of plausibility in section §6.4 of chapter 2 involves inference and so we will need to be able to represent inference rules within the system.

Notice also that the types of structure which these phenomena require will be recursive; beliefs about beliefs, discourses about beliefs and beliefs about rules of inference are quite normal.

There are at least three ways of achieving this extra structure[2].

Located values The value assigned to a term by a description could contain the extra information. In such a system the value associated with a term by a description might be a set of pairs with the first element being a location (for instance "Mary's beliefs yesterday") and the second a simple boolean value.

Structured terms The terms could be more structured. Instead of a simple combination of a functor and some entities it could be a

---

[2]Here we consider only fairly direct representations. See section §3.3 of chapter 3 for a discussion of KAMP and how its more indirect representation affects its planning behaviour

larger object including the extra information needed. A term in such a system would represent something like "Mary believes that John believed yesterday that...".

Multiple descriptions A state of the world could be described by a collection of simple descriptions of the kind described in section§3.3. There would have to be some way of indicating the structure required in addition to the descriptions themselves.

We choose to use a variant on the third alternative, since it makes the extensions described in section §4.3 which deal with equality and disagreement about the identity of entities simpler. In section §5.3.2 we discuss some situations in which we need to be able to deal with structures more like structured terms and how these can be extracted from a representation in terms of multiple descriptions.

The structuring required between descriptions could be provided in a number of ways; the basic need is that it be a directed graph, in order to provide for recursive structure and sharing of structure (as, for instance, when two agents are assumed to be manipulating the same 'discourse context' in a shared dialogue).

Such a structure might be used to represent the following situation —

> John believes he is sitting in his car, which is blue, and smoking. John also believes Mary shares this knowledge and is saying "hello".

by building a graph where there are labelled arcs between descriptions which represent the relationships between agent's beliefs and the discourse situation. So, let us say that we have decided that the following arc labels are meaningful

Mary-believes Arcs with this label link descriptions of someone's beliefs with the description of what they believe Mary believes.

John-believes Similarly this indicates beliefs about John's beliefs.

d-context An arc with this label links a description of someone's beliefs with a description of the discourse in which they are engaged.

Using these definitions we could build a representation for the situation described above as shown in figure 5–6. The incoming arrow is intended to indicate that the

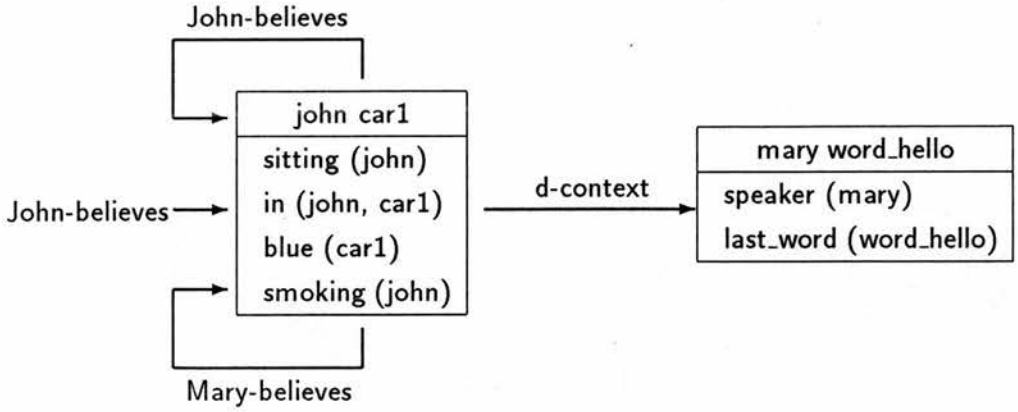


Figure 5–6: Belief Structures Described as a Labelled Di-Graph

left hand description describes John's beliefs. The topmost arc simply represents John's knowledge of his own beliefs, since by the definitions just given it says that John believes that John's beliefs are as described in that same description. The bottom arc says the same about Mary, so, by asserting that this structure describes John's beliefs we are asserting the following infinite set of terms —

John believes that 'sitting (john)' is true.

John believes that John believes that 'sitting (john)' is true.

John believes that Mary believes that 'sitting (john)' is true.

John believes Mary believes John believes 'sitting (john)' is true.

The right hand description represents some terms about the discourse in which John and Mary are involved; here for the sake of the example it is fairly simple, just noting who is speaking and what she is saying. The representation asserts that John believes the discourse is in that state and that John believes that Mary believes that the discourse in this same state and so on. Of course John might be mistaken about Mary's beliefs about the discourse if, for instance, he had misheard what she said.

Notice that we have no representation of Mary's beliefs here, only of John's beliefs about them; this representation is 'subjective' in the way described in section §2 with John as the special agent whose part the system is playing.

This representation seems to satisfy many of our requirements. However the goal of having the system be 'self describing', as put forward in the introduction to

this chapter, means we should be wary of introducing extra layers of complexity to the representation in this way when a simple generalisation of what is already present will give the same power. We have already provided for facts to have different values in different descriptions and if we allow values other than simply True or False, then we can describe the relationships between different descriptions by introducing functors in place of the arc labels of the graph representation. For instance we can represent the structure above by defining two new functors 'd_context' and 'believe' and building facts 'd_context ()' (a fact having no entity components which in future will be written simply as 'd_context'), 'believe (mary)' and 'believe (john)'. We then assign them descriptions as values indicating the structure we require. We refer to such facts as 'modal facts', borrowing the name from modal logic whose concern is precisely such highly structured descriptions [Hughes and Cresswell 1968]. This might be seen simply as moving the arc labels inside the descriptions where they can be directly manipulated.

Writing out such a structure causes problems since a description can contain a term which has as its value that same description — as indeed is the case in the above example. To get around this we add to each description an arbitrary name and in those cases where it would be necessary to write out a description in a place which is inconvenient we write an empty description with the same name. Using this convention the above example can be portrayed as shown in figure 5–7. Where the values of the 'modal' terms are written just below them. It is this kind of structure, rather than the network representation, which will form the basis for further elaborations in the coming sections.

§4.3 Describing the Beliefs of Others

The modal terms, given descriptions of the world as values, introduced in the last section provide the basis of a way of representing the beliefs of other agents, as was shown in the example with 'believe (mary)'. However there is a problem with which it does not cope.

The essential reason for the system to represent the beliefs of other agents is that they may disagree about the state of the world, indeed this is one of the most common reasons for an agent to need to engage in communicative behaviour. A representation of the kind outlined above can represent differences in the assignment of truth values to terms and similar differences in modal contexts such as

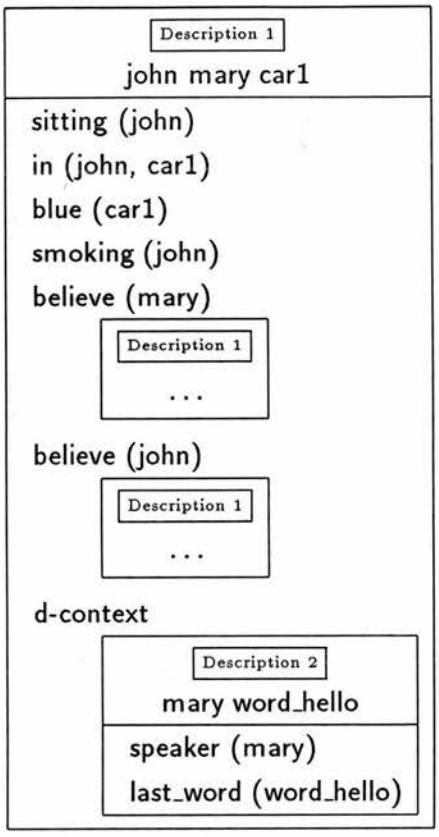


Figure 5–7: The same state using modal facts

disagreement about the beliefs of a third party. It cannot, however, represent differences as to the number and identity of entities. For instance take the following description

> Mary is addressing a police officer whom she, mistakenly, believes to be Kate. John and Mary both know that Kate is a violinist.

Let us try to represent the situation as it is just before Mary makes her, incorrect, identification. It might be as shown in figure 5–8. For clarity we ignore the agents beliefs about their own beliefs and so on. At the moment here there is no disagreement, John's beliefs about Kate and the police officer are exactly the same as Mary's. However, when Mary decides that the police officer is Kate then there will be two major disagreements.

- Mary will believe that the police officer is a violinist.
- She will believe the violinist is a police officer.

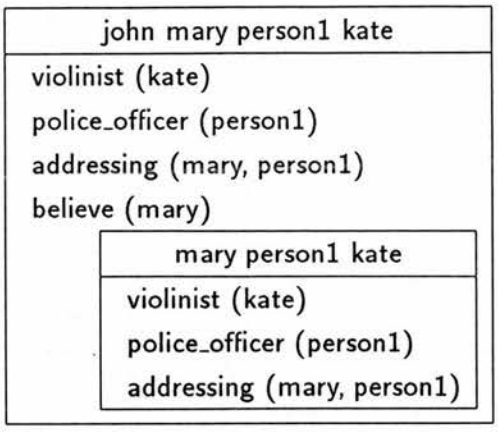


Figure 5–8: The world before Mary makes a mistaken identity (from John's point of view)

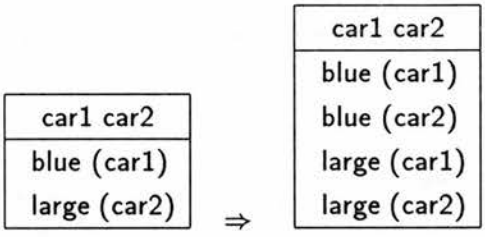


Figure 5–9: The result of asserting car1 = car2 in a description

In addition, if we were to add information to the state description to indicate what names John and Mary believe to belong to each entity in their model of the world we would find that Mary believes that the person who is named "Kate" is both a police officer and a violinist.

How would such a change be made in the representation given here? We could simply rebuild John's beliefs about Mary's beliefs, but such a reconstruction would be costly in a more realistic situation where Mary had many beliefs about Kate, all of which would need to be transferred to the police officer. For instance figure 5–9 shows the transformation which would need to be made when adding a single equality 'car1 = car2' to a small state description. Worse, if the original context had been that shown in figure 5–10, the resulting state description would have been inconsistent, assigning both True and False to 'blue (car1)' and 'blue (car2)'.

An alternative would be to represent the new identity within the representation. In a standard logic this would be done by identity formulæ such as

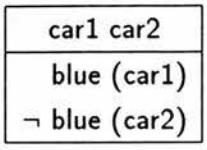


Figure 5–10: A state description which becomes inconsistent

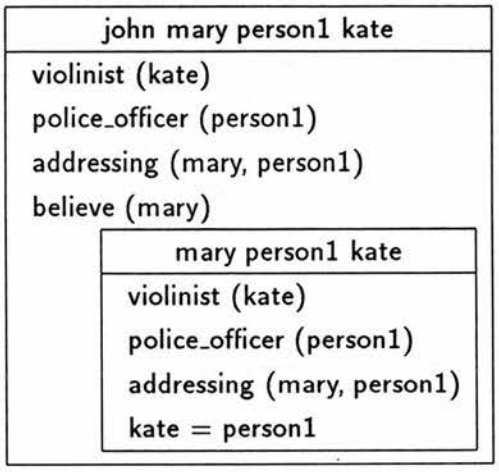


Figure 5–11: Representing equality explicitly

Kate = person1

If we were to try to do this within the current representation we would get a description like that shown in figure 5–11. However the '=' term has a non-obvious interpretation and, worse, can not be manipulated in the same way as any other. Changing the value of such terms can have a large effect on the meaning of a description and it becomes more complex to answer simple questions about the states represented by a given description since we must look through the whole description to find any identities and apply them all to each term which is given a value in order to determine the full set of constraints which the description places on the state. We stated in section §3.3 that determining the value of a term in a description was simply a matter of looking it up, with no deduction, in keeping with the decision noted in chapter 3 to develop a passive world model leaving all manipulations to the planning algorithm. If we are to represent identity explicitly in this way then we must abandon this simple evaluation strategy.

A solution to these problems, and others which arise from the philosophical problem of "cross world identity" has been developed by Fauconnier [Fauconnier 1985]. Here we shall take from that work the central idea that a model of the world, such

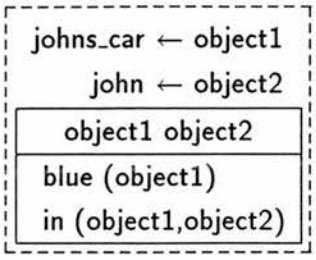


Figure 5–12: A Simple View

as we might use to describe an agent's beliefs, should be separated from the description of how the entities in the model correspond to entities in other models.

Consider the following formulation of what happens when Mary makes the identification of Kate and the police officer. Mary's model of the world changes to be one where there is only one entity which corresponds to both her concept of Kate and of the police officer in her previous beliefs. From John's point of view a similar thing can be seen to be happening; John views Mary's beliefs in such a way that Mary's single entity is seen by him to correspond to both of his concepts. These correspondences (both for conflation of existing entities and for interpreting the concepts of other agents as corresponding to an agent's own concepts) can be achieved by separating the representation of individual entities in a description from their interpretation. To do this it is simplest to ensure that the entities in any two descriptions are distinct so that there are no accidental conflations and then refine our notion of 'modal facts' so that their value is defined to be, not a description, but a pair consisting of a description and instructions on how the entities in the description are to be interpreted. This interpretation information will just be a set of mappings giving, for each entity in the description, a counterpart in the surrounding description. The combination of a description and the interpretation mapping we call a 'view'. A simple view is shown in figure 5–12. In general the mapping between entities might be an arbitrary relation between the two sets of entities. Mappings where more than one entity in the inner description correspond to a single entity in the outer one can be used to represent states off affairs where a single entity plays more than one rôle, as happens for instance in Mary's confusion of Kate and the police officer. Mappings where one entity in the inner description maps to more than one entity in the outer one would represent confusions of the opposite kind, for instance where an agent believes that there are two identical entities in the world when in fact there is only one which they have seen in different circumstances. [Fauconnier 1985] contains many interest-

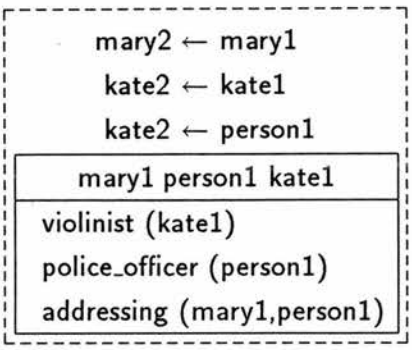


Figure 5–13: A View Representing Mary's Identification of Kate and the Police Officer

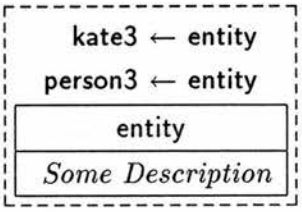


Figure 5–14: The Form of John's Beliefs About Mary's Beliefs

ing examples of confusions of various kinds and discusses how this separation of description from identity of entities can be used to represent them.

The description which forms part of the view should be seen as a very abstract description of a state of affairs, in this case the state of some entity being inside some other which is blue. The mapping written above it shows how this abstract description is to be interpreted in terms of actual entities, here John is in his car which is blue. This is obviously similar to binding variables in a unification based system and the mappings will be useful for this purpose later.

We can use a view to represent Mary's new belief in terms of her past beliefs as shown in figure 5–13 with one new entity 'Kate2' given as corresponding to two in the old beliefs.

In a similar way we can see that John's beliefs about Mary's new beliefs will be such that he sees Mary's one entity as corresponding to two of his own, in a view of the form shown in figure 5–14.

Our full representation of the situation must take into account both changes — the change in Mary's model of the world and the change in John's interpretation

a ← m Combined With m ← x Gives a ← x	a ← m b ← m Combined With m ← x Gives a ← x b ← x	a ← m b ← m Combined With m ← x m ← y Gives a ← x a ← y b ← x b ← y

Figure 5–15: The Result of Combining Mappings

of it. To do this we need some rule to combine mappings. Here we use a simple concatenation which gives the results shown in figure 5–15.

Using this combination rule the complete representation for the example of John, Mary and Kate, after Mary has made the incorrect identification of the police officer and the violinist, would be that shown in figure 5–16. The double mapping of both John's concept of Mary and of the police officer ensures that John is able to assert things such as

> Mary believes that Kate is a police officer.

which he obviously should in this situation.

A final example is the following situation —

> John is sitting in his car, which is blue, and smoking. Mary shares all these beliefs and is addressing a policeman who she believes to be John.

which is described by the state in figure 5–17. Notice here how the same description ('description 4') can represent different beliefs sets depending how the entities it refers to are interpreted.

§4.4 Summary of Representation

The representation scheme for states of the world which has been developed in this section is as follows.

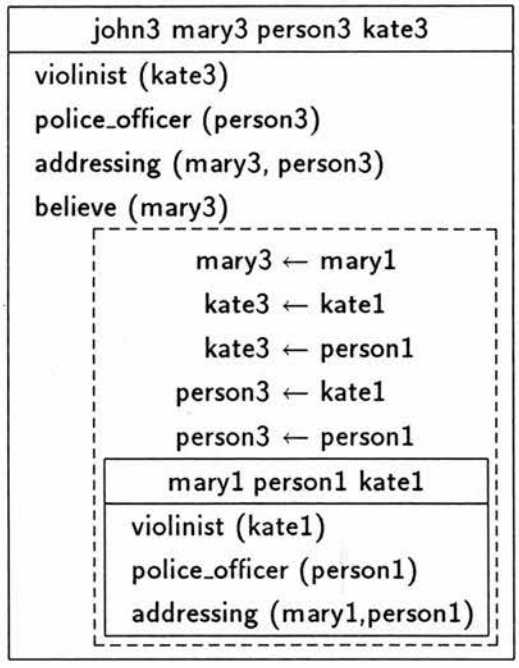


Figure 5–16: The World from John's Point of View After Mary's Mistaken Identification

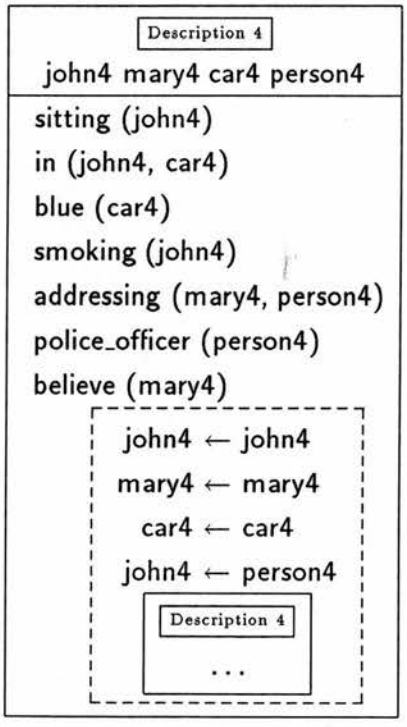


Figure 5–17: A Description With Mutual Belief

- The representation is built up of simple descriptions which assign values to facts.

- Associated with each description is a set of entities which are assumed to exist in the world as described by the description.

- Modal concepts are represented by giving facts values which consist of a description and an interpretation of the entities in the description in the form of a correspondence between entities known and entities in the description.

This representation allows for the creation and examination of world descriptions and for the representation of multiple descriptions of the world which differ in terms of the value assigned to facts and even the number and identity of the existent entities.

§5 Plans and Planning

It was argued in chapter 3 that language usage should be seen as just another kind of action and that a language-using computer system should be based in a model of rational action, that is planning. In this section we develop a representation of plans and a planning algorithm which will be described in more detail in chapter 6 and described formally in appendices B and C.

One criterion which has been applied repeatedly in designing this system is that of 'self description', as defined in the introduction to this chapter. Although it has not been an aim of this work to produce a planner capable of representing its own operations, it was considered worthwhile to make choices in defining the representation and mechanisms used in such a way as to minimise the obviously non-representable features. Thus the structures described in section §4 above for use in representing states of the world and people's beliefs are also used to represent states in the plan and the relationships between them.

§5.1 A Basic Planning System

We will present our system in two stages. In this section we present the core of the planning process. Most of this consists of well known structures and mechanisms from the planning literature adapted to the specific needs of text planning.

Sections §5.3 to §5.4 will extend this model to cope with some of the more difficult properties of this domain. In all of the following we assume a very narrow notion of what constitutes a planning system, keeping within the passive world model, structural planning paradigm decided on in chapter 3.

Traditionally, planners have been seen as systems which are given a state of the world and a set of aims, or 'goals', and produce a plan which is believed to achieve those aims. Actually performing those actions, executing the plan, has been seen as a task to be performed by a separate, though related, system. When it comes to systems which must take into account the actions of other agents, and even more systems which must be able to deal with the potentially infinite number of tasks which must be performed to decide on the full consequences of giving some piece of information to an agent capable of deduction, it becomes necessary to interleave the construction of a plan with its execution so that it is not necessary to have a complete plan before any action is taken. It is also necessary to abandon the notion that an agent can be seen as having a fixed set of goals which it is to achieve and then halt; an agent which is to work in the real world must be able to derive its own specific goals from more general considerations and the current state of affairs. Furthermore, any system which is to interact with others must be able to cause them to adopt new goals in order to modify their behaviour and so must have a model of goal production.

These considerations lead to a notion of a planning system whose basic action is to repeatedly do one of three things

- Extend its plan.
- Perform an action which the plan indicates should be done immediately.
- Derive new goals for itself or for other agents.

It is this kind of system which will be described in more detail in the following sections.

An operating planning system can be seen as consisting of at least the following parts —

World State A description of the current state of the world.

Current Plan A description of the course of events which the system is hoping to cause to happen in the future.

Actions A set of possible actions, or action types, each of which is described in enough detail for the system to be able to determine what must be done before it is performed and what the result of the action should be.

Agenda The set of things which the planner has determined that it must do. This might include goals which the system has determined should be achieved, unresolved problems with its plan and/or actions which must be performed.

Constraints A set of constraints which future manipulations of the plan must not be allowed to violate.

A number of additional parts must be included in a system to perform actions in a domain involving communication; these will be introduced in later sections when we come across problems which this model does not address.

§5.1.1 Current World State

The world state can just be described by a 'view' of the type described in section §4.3. The manipulations described there can be used to change the state when necessary.

§5.1.2 Current Plan

The most common representations for plans put forward for AI systems are —

- A simple list of actions with information about the state of the world before and after each.
- A description of a sequence of events in some form of temporal logic.
- A network of actions representing a partial ordering in time.

Each of these has drawbacks for the kind of system being described here.

A simple list of actions forces the system to commit itself to the precise ordering of each planned action when it is first inserted into the plan. Apart from the extra search this may involve, this is really not appropriate when some of the actions in a

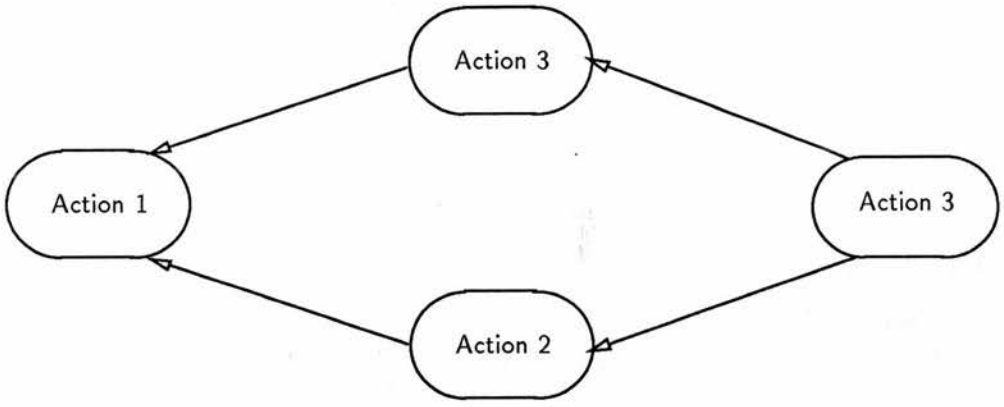


Figure 5–18: A Network Of Actions

plan are to be executed by other agents and their ordering may not be important. An example of such a situation in a language using system arises when the system must plan for its audience to perform some inferences; it may not matter precisely in what order the inferences are made, so long as they are performed before the system makes its next utterance.

Temporal logics are a very powerful representations, and it would be possible to represent more or less any variety of plan or constraint in this form. However, the resulting representation is not at all transparent and asking questions about the plan would involve potentially arbitrary inference.

Plan networks allow for the representation of partial information about the order of actions. However, some quite simple manipulations on such networks involve looking at all of the plan, when they would seem to be purely local in effect.

As an example take the network of actions in figure 5–18, where the arrows represent temporal constraints on the actions indicating that action 2 and action 3 must both take place after action 1 and that action 4 must take place after both action2 and action3. If we wish to constrain 'action 3' to be after 'action 2' then we must also delete an ordering link from 'action 4' to get the network of figure 5–19 rather than that of figure 5–20. Since the usual algorithms for determining what is true at a point in a plan (e.g. [Tate 1976]) demand that it contains no redundant links and would be deceived by the link from action 2 to action 4 into thinking that states brought about by action 2 will still be the case when action 4 is performed, no matter what is changed by action 3. An alternative to deleting redundant links is to complicate the algorithm for querying facts about a plan,

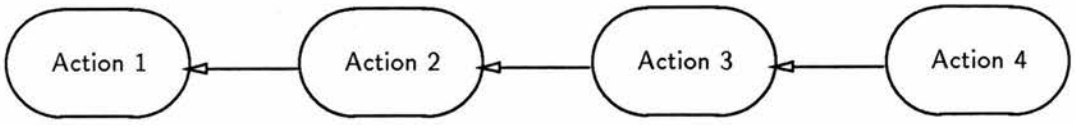


Figure 5–19: Correctly Linearised

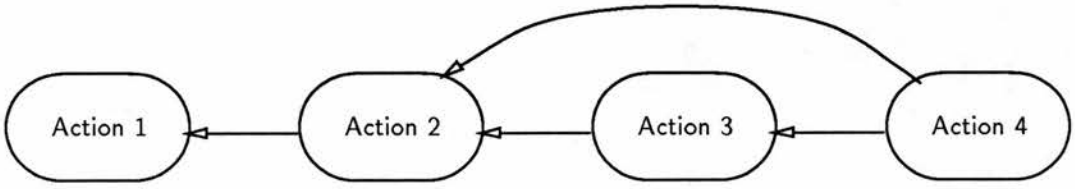


Figure 5–20: Network with a redundant Link

which would make the system much less efficient — especially when the queries to be made are complex as would be the case when the plan represents the beliefs of more than one agent — and also harder to describe in the planning formalism, as we would want for 'self description'.

To avoid this problem we can make the observation that what the system is really interested in is the state of the world and how it changes over time. The links in a network of actions are really used to force the changes which the system plans to make to the state of the world to happen in a fixed order. A corollary of this is the ordering of the actions which cause the changes. If we represent these different states of the world we can simplify the rule that there should be no redundant links to simply saying that each state has one start point and one end point. As was mentioned in chapter 3, dealing with time periods rather than time points also has the advantage of allowing the system to use the same language to talk about the ordering of actions and restrictions on what can be allowed to change. This is, of course, not a new idea; for instance see [Steel and Leung 1989, Allen 1984b, Cheeseman 1983] which describe similar systems and discuss various ways of manipulating the time periods.

Now we have already designed a very versatile type of description for states of affairs in section §4.3 where we called it a 'view'; a desire for overall simplicity as well as the wish for a system able to describe its own operation, as stated in the introduction to this chapter, would lead us to using that same form of state description to build our plans.

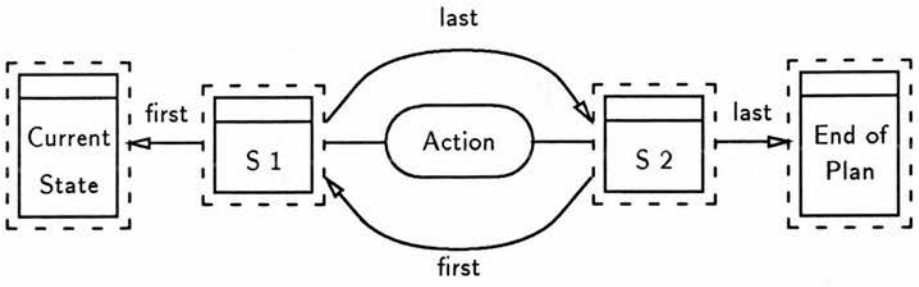


Figure 5–21: A Plan of One Action Constructed of Views

Using views as the basic building blocks, a plan of one action might be represented as shown in figure 5–21. Here the rectangles represent views and the ellipse an action which occurs between the two states 'S1' and 'S2' (for the rest of this section we omit the dotted line around views but it should be assumed that the state descriptions which make up a plan are views not simple descriptions).The arrows are to be read to say that 'S1' holds from immediately after the current state ceases to hold to immediately before 'S2' does. Similarly 'S2' holds from immediately after 'S1' ceases to until just before the end of the plan. That is to say, the state descriptions and their associated 'Earliest' and 'Latest' indicators define an open interval of time over which the view is a true description of the world.

The fact that the results of the action are asserted to be in force immediately after the preconditions is a result of the "No Parallelism" assumption noted in section §2.

Using this notation the three action plan segment shown earlier can be represented by the network of views in figure 5–22. Notice that this diagram indicates that the state of the world which forms the precondition to 'action 4' holds from after 'action 1'. This is unlikely, since if it was the case there would have been no reason for 'action 2' and 'action 3' to have been added to the plan. In fact a real plan would be structured as shown in figure 5–23. (assuming that 'action 2' was inserted into the plan before 'action 3' and the two actions do not interfere in any way.)

§5.1.3 Actions

Given the plan representation described above we can go on to decide how the system should represent actions. From the examples at the end of section §5.1.2 it

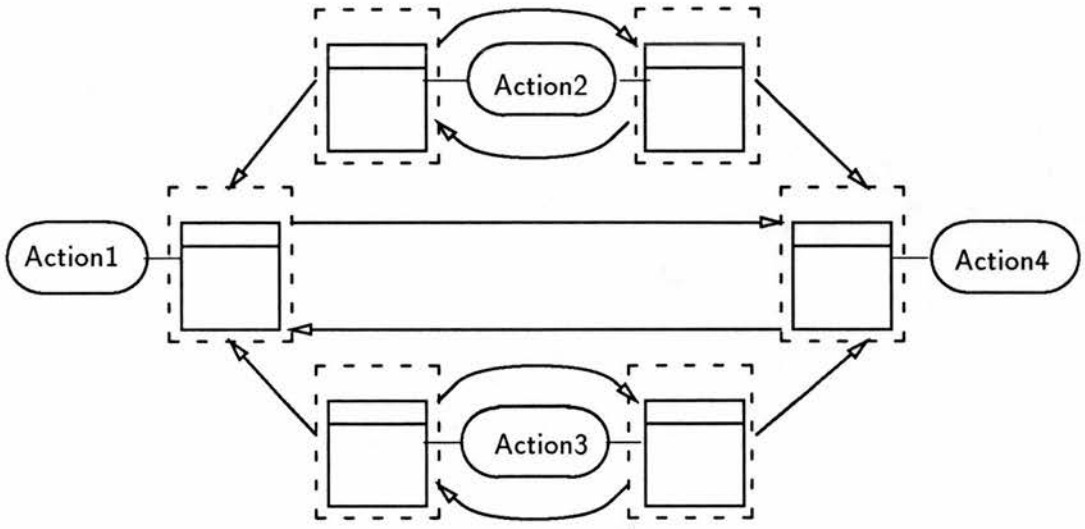


Figure 5–22: The Three Action Plan Built of Views

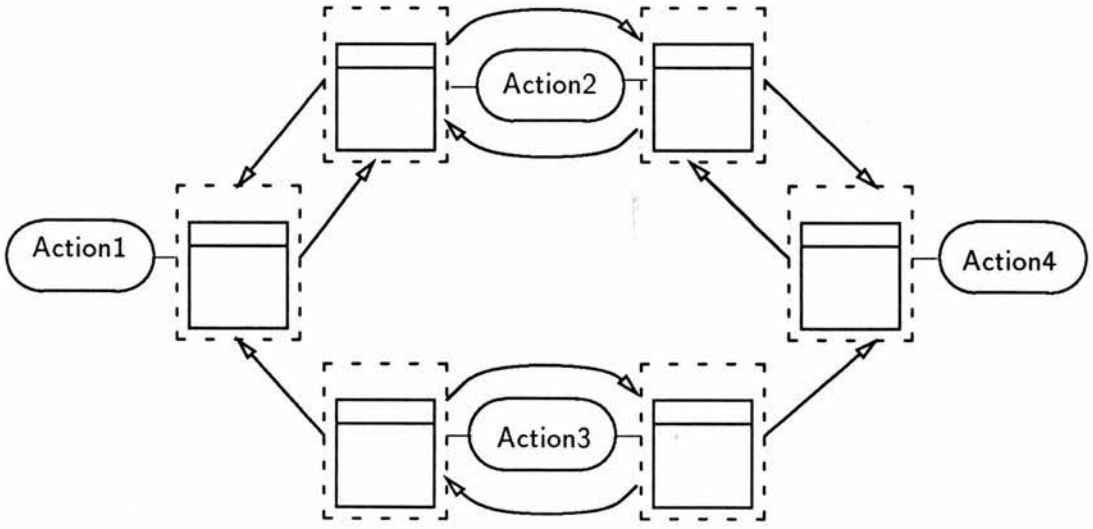


Figure 5–23: Corrected Three Action Plan

can be seen that the essential parts of an action from the planner's point of view are two descriptions. The first describes the minimum conditions on the state of the world which will allow the actions to be executed. The second describes the minimal conditions which can be assumed to be the case after the action. This corresponds to the preconditions and the add and delete lists of a Strips type planner respectively. In addition we will include in the action a description which can be seen as the 'name' of the action as it would be seen by outside processes, though this will have no meaning to the planning process itself.

Usually a planner will not want to have a representation for every action which it can perform since there could be a large, or even infinite, number. The number of actions which need be represented is usually reduced by introducing pseudo-entities which behave like logical variables in that they are used as place holders for any of a large number of entities.

For instance we could avoid representing the actions

Move block1 to block2
Move block1 to block3
Move block2 to block1
⋮

and so on by instead representing a single action type

Move ?A to ?B

where '?A' and '?B' are place holders[3]. We must also specify that '?A' and '?B' are standing for blocks rather than whatever other entities might exist in the planner's universe.

---

[3]We use the term "place holders" rather than the more common "variables" so as not to imply that there is any difference in type between entities which behave as place holders and those which do not. The difference lies not in the entities in themselves, but rather in the amount of information which we have about them. Entities which are acting as place holders have not yet been identified with a known object in the 'real world' and so an action containing place holders may not be executed until such an identification is made. See section §5.2.3.4

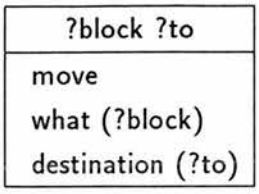


Figure 5–24: A Simple Action Description

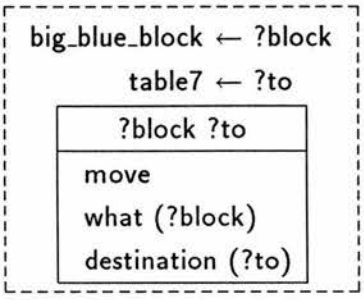


Figure 5–25: A View of The Action Description

When the planner comes to use such action types it has to replace the place holders with the actual entities which are to be manipulated in the plan which it is constructing. It must do this replacement not just in the action's description but also in its pre-conditions and post-conditions.

As was shown in section §4.3, the mappings which form part of a view are designed to allow for situations where one entity is to be interpreted as standing in place of another. The representation of the pre- and post-conditions as views makes it simple to do the binding of the place holders to specific entities. What is needed is a way of describing an action which shares the same properties.

We do this by defining a description of an action similar to the descriptions of states presented in section §4.3 and adding a mapping to it in the same way as was done there to get views.

An action description then will consist of a number of terms, each consisting of a functor and a number of entities. An example of such a description is shown in figure 5–24. This action description might be used to represent the class of actions which involve putting something in some place. The choice of functors used to build these action description terms is, of course, totally arbitrary and the only reason to choose a description in three terms rather than a single term such as move (?block, ?to) is readability.

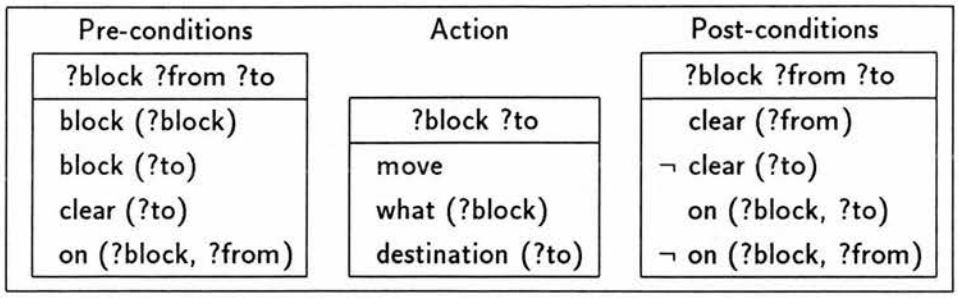


Figure 5–26: A Complete Action Description

When we need a particular instance of this type of action, with a particular block being moved to a particular place, we need only add an interpretation for the entities '?block' and '?to' to the above representation, producing something like the description in figure 5–25. When it is inserted into a plan this would be interpreted as representing a specific action, that of putting 'big-blue-block' on 'table7'.

To summarise, an action is represented as a triple consisting of a view describing the conditions under which the action may be performed, a view describing the type of action and the entities involved and a second view representing the conditions which can be expected to hold after the action has been performed. A complete representation of the 'move' action type which was described above might be as shown in figure 5–26. Notice that the corresponding entities in all three views are the same. The underlying contexts would, as always, have independent entities, but the mappings which are added to make the views will force them to be interpreted correctly.

§5.1.4 Agenda

The system's agenda can simply be seen as a list of "things to do", which we call "chores", which is kept in an order determined by priorities which reflect the urgency which the system attaches to them. Obviously there are many ways of organising such an agenda but here we will simply assume that some priority is assigned to the chores so that the system knows which it should tackle next. We discuss the agenda system used in the actual program Riple in more detail in section §4.3 of chapter 6.

A distinction is made here between the 'goals' of the system which are those states which it means to bring about in its world, and the 'chores' which are placed on

the system's agenda. The latter are steps in the operation of the system, notes of formal manipulations of the plan and other data structures of the system, while the former are the aims of the system with respect to outside conditions. The similarity of these two concepts arises from the fact that both the goals of the system and the chores of the planning algorithm can be seen as corresponding entities in different domains. The planning algorithm can be seen as a rather inflexible solution to the problems which arise in a domain consisting of 'actions', 'states', 'agents' and so on; indeed it should be possible to produce a system which solves problems in both its external domain and the internal, planning, domain using the same strategies. It is the desirability of such single layer systems which is the motivation for the target of "self description" put forward in section §2 of this chapter and used to guide a number of the decisions made in the design of the system being described. Some work on single level systems has been done (e.g. [Wilensky 1983], [Bartle 1984]); however no attempt to actually produce such a single level system has been made in this work and there would be large problems to be overcome in the construction of such a system based on that described here.

The chores which the system must perform as it operates break down into a number of classes. Among them are —

Support an Unsupported Fact Any fact which the planner has decided must be true at some point in the plan must be 'supported'. That is the planner must be certain that no matter how it decides to manipulate the plan at a later stage the desired state will be true at the desired point.

Eliminate a Conflict If two states in the plan are indicated to have overlapping bounds and assign different values to a fact then they are said to be 'conflicting'. For each pair of conflicting states the system must change the plan so that they no longer conflict.

Bind a Place Holder Some of the place holders in the actions in the plan will have been associated with a specific entity ('bound') when the action was chosen. Others might be bound at a later time in order to supply support for some goal. The remaining place holders must be bound before the action is performed since even if it is unimportant what it is bound

to some choice must be made. For instance it may not matter where a block is put when it is being moved to make room for another but it must be put in some specific place. Thus some of the chores which the system must perform will consist of choosing which entity each otherwise unspecified place holder will represent.

Execute an Action Every action which is inserted into the plan must be executed at some point consistent with the constraints which the structure of the plan places on it.

Others will be introduced later as the planning model is extended.

§5.1.5 Constraints

As choices are made about how the plan is to be extended, there will occur situations where it is necessary to place constraints on future manipulations of the plan so as to ensure that the improvements made are not undone by future manipulations. An obvious case is when an action is inserted into a plan in order to support a goal; in this case the system must be sure that future changes in the plan do not cause the goal to lose its support.

There are two types of constraint which are needed —

- Sometimes the beginning and end points which a state has been given in a plan must not be changed. A good example is when two actions have been forced to have a fixed order so that they do not conflict; in such a case the 'after' state of the first will have been placed earlier in the plan than the 'before' of the later and this must not be changed.

- Sometimes it is necessary to ensure that a certain condition will hold over some time period, no matter what changes are made to the plan. This is the case when an action has been introduced to support a goal; the goal must be held consistent from whenever the action is performed (which might change as the plan is further developed) and when the goal is required.

Constraints of the first kind can be handled by simply marking some of the 'first' and 'last' relationships in the plan to be fixed and immovable. The second kind

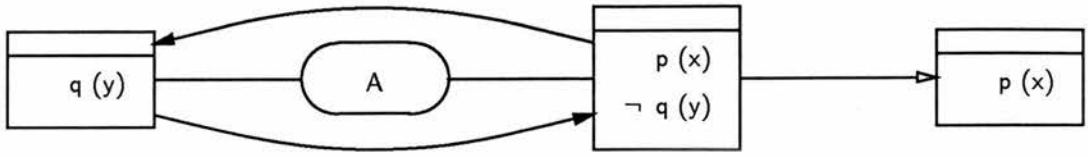


Figure 5–27: A Plan Fragment With Fixed Links

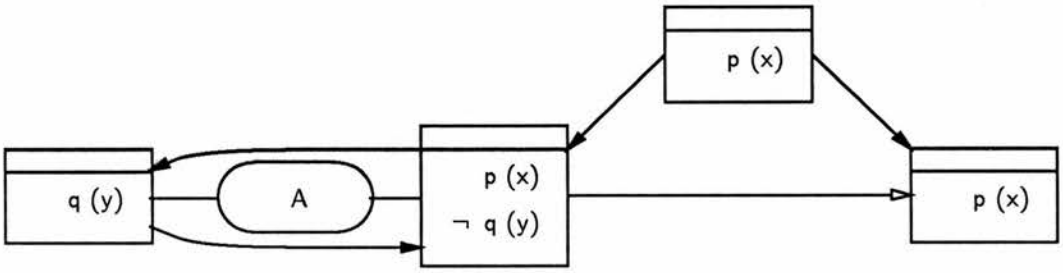


Figure 5–28: Inserting a State to Enforce a Constraint

of constraint is more complex, but it can be quite simply represented by inserting extra states into the plan indicating the period over which the constraint holds.

For instance, take the plan fragment shown in figure 5–27 where the action 'A' has been inserted into the plan to support the goal 'p (x)'. (here I have indicated the fixed relationships with fixed arrowheads — the pre- and post-conditions of the action must occur one after the other, with no intervening states, according to the 'No Parallelism' assumption described in section §2.) The problem with this fragment is that the relationship between the action and the goal might be upset be later changes in the plan, so invalidating the support. One solution would be to make this relationship fixed like those between the before and after states of the action. This would certainly stop the system from removing support for 'p (x)', however it would place much too strong a constraint on the plan since it would force '¬ q (y)' to be true over the same period. A second option, the one we take here, is to add an extra state to the plan as shown in figure 5–28. Here the extra state with its fixed relationships ensures that exactly the required state holds over the period from the performance of the action to the requirement for the goal to be supported. The system is totally free to change the plan so that 'q (y)' is reasserted in the intervening time if it needs to.

§5.2 The Operation of The Basic System

A planning system of the kind described in section §5.1 would be started by giving it a description of the current state of the world and a number of action types which it may perform. Possibly it might also be given an initial plan in which case its agenda must be initialised to contain the chores to which that plan gives rise (see section §5.2.1 for a description of how properties of a plan can give rise to chores to be performed), otherwise the agenda would initially be empty .

The basic behaviour of such a system is like that of any agenda-based problem solver, it will repeat the following sequence of steps

Chore Identification Examine the current plan and determine what needs to be done and add these chores to the agenda.

Choose a Problem Select a chore from the agenda to be performed.

Choose a Fix For each type of chore there will be a number of possible methods of performing it; at this stage we choose one of these.

Perform Fix Perform the fix. This will change the plan, the state of the world and/or the constraints.

When there is nothing on the agenda at stage 2, then the planner can be said to have successfully completed its task. When there are no fixes available at stage 3 then the planner fails in its task. In most cases the system's task will be open-ended and so neither of these things will occur.

One obvious problem with the planning behaviour given here is that it will only work if the system can choose the correct 'fix' for each chore. There are times when the choice of fix is important, for instance choosing to refer to an entity with a pronoun will prevent latter addition of a relative clause. In such cases the system needs to be able to try a different fix for a past chore and try again when it finds itself unable to proceed at stage 3. The method by which we choose what chore to redo can range from simply reverting to the last time we had a choice ("chronological backtracking" as it is called) to intelligent systems which know how to undo a chore without disturbing later choices. In the system described here we choose the simplest chronological backtracking scheme (limited by the

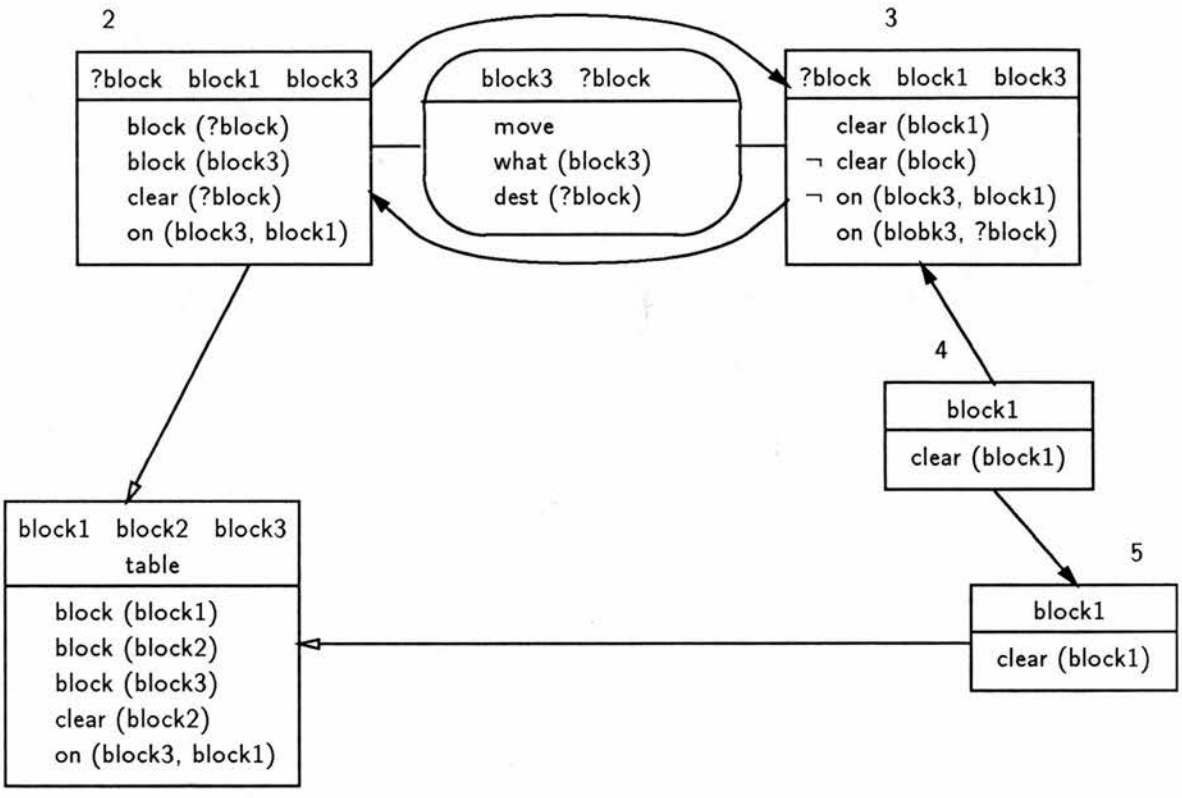


Figure 5–29: An example plan

fact that the system cannot backtrack over a choice to execute an action, since it has changed the world, see section §5.2.3.5) so as to avoid problems which, though important for efficient and practical systems, are not directly relevant to the problem of planning discourse.

The following subsections discuss each of these steps and describe what they involve for the basic system described in section §5.1; the examples will be based on the plan fragment shown in figure 5–29. We assume here that the 'move' action, consisting of states 2 and 3 together with the action description shown here between them, has just been inserted into the plan in order to provide support for the fact '**clear (block1)**'; the state labelled 4 with its fixed bounds was added at the same time — see section §5.1.5 for an explanation of why this was done. The place holder '**?block**' has been left unbound since it does not matter where the block '**block3**' is placed.

§5.2.1 Chore Identification

In the basic system there are 4 kinds of chore:

Provide Support Every fact which is asserted to be true in the plan must be known to be supported in some way, that is the system must know what action or constraint in the plan is responsible for their holding. To ensure this as facts are inserted into the plan support must be found for them. This is how the system's goals are attained, goal states would be inserted into the plan forcing the system to ensure that they are brought about.

Eliminate Conflict Any conflict in the plan must be removed.

Bind Place Holder Place holders which were left unbound when an action was inserted into the plan must be replaced with definite entities at some point.

Execute an Action When manipulation of the plan makes the view which defines the preconditions of an action be that view which describes the current state of the world, then the action should be executed.

Make a State Current Any state all of whose constituent facts are supported and whose 'earliest' time is marked as being the current state of the world must be merged into the current state of the world.

The system can simply examine all changes which were made to the plan at the last iteration of the steps described in section §5.2 looking for new chores of each type to be added to the relevant agenda.

In the example three states have just been inserted into the plan, numbered 2, 3 and 4 in the diagram, and the facts listed in figure 5–30 are not known to be supported.

There is no state which fulfills the conditions to be 'made current' and no action which can be executed. Just one place holder, '?block', needs to be bound and so this chore is placed on its agenda.

block (?block)	at 2	clear (block1)	at 3
block (block3)	at 2	¬ clear (?block)	at 3
clear (?block)	at 2	¬ on (block3, block1)	at 3
on (block3, block1)	at 2	on (block3, ?block)	at 3
clear (block1)	at 4		

Figure 5–30: Unsupported Facts in the example Plan

§5.2.2 Choose a Chore

In section §5.1 we did not specify the organisation of the system's agenda of chores. In an actual planning system we there are many possible organisations, but for the purpose of this explanation we do not need to be specific. In fact for ease of explanation we shall examine the various chores which the system must perform in section §5.2.3 without regard to the priority which the system assigns them.

§5.2.3 Choose a Fix

Having chosen a chore the system must choose a fix for that chore. First of all it must check that the chore has not been fortuitously fixed by changes to the plan since the chore was detected, this can be done simply using the same tests which are used for chore detection.

There are a number of ways to fix most of the problems which might be tackled by the system. In addition there will be a number of variations on each type of fix — for instance a place holder can be bound to one of a number of entities. The possible fixes for each type of chore are described in the following sections.

§5.2.3.1 Finding support

The chore for which the largest number of types of fix are possible is finding support. The following types of fix are possible.

State of the World All facts asserted by the current state of the world are supported with no further work by the system.

After an Action A state which is the post-condition of an action is supported by that action since the definition of the action is that it will cause that state to come about. In this case the system need do nothing.

In a Constraint States which are inserted as constraints (rather than, for instance, as part of an action) are guaranteed support by the insertion.

Made True Earlier If the fact is asserted earlier in the plan then it can be supported by inserting a constraint which assures that the fact will remain asserted between its earlier assertion and the state needing support. This constraint will take the form of a single fact state, like 4 in the example, which will have its bounds fixed as the earlier assertion and the state needing support.

Can Make Correct If the fact requiring support can be made to have the correct value by one of the action types which the system has available, then an action of that type can be inserted into the plan to give the required support. The 'First' link of the pre-condition of the action will be attached to the beginning of the plan and the 'Last' link of the post-condition to the end of the plan since the system knows no better than that the conditions will last that long. In addition a constraint will be inserted to ensure the newly supported fact will remain supported. This is the type of fix which had just been performed to get the example plan fragment; the 'move' action consisting of states 2 and 3 and the description between them had been inserted and the constraint, 4, placed between the post-condition and the place where support was required.

It must be noted that any of the above types of fix must be selected taking into account the presence of place holders since when the fix is applied the place holder may need to be bound.

In the example, the following facts in state 3 are supported by the action

clear (block1)	at 3	¬ clear (?block)	at 3
¬ on (block3, block1)	at 3	on (block3, ?block)	at 3

and the following in state 4 is supported by being in a constraint.

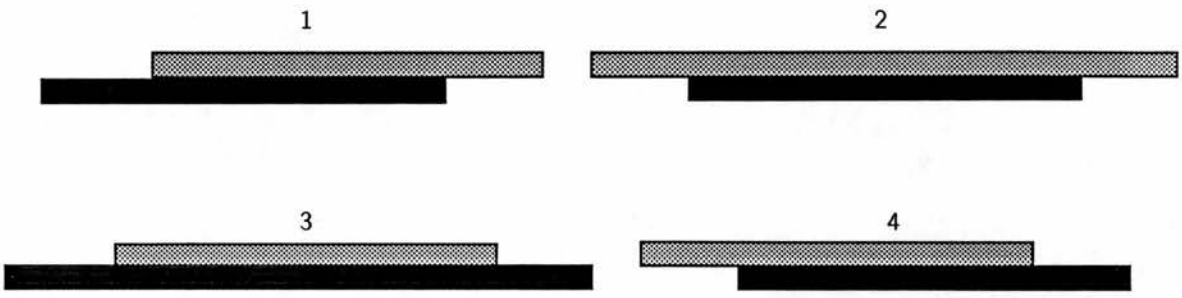


Figure 5–31: Possibilities for temporal overlap Between States

clear (block1) at 4

This leaves the following facts in need of support

block (?block)	at 2	block (block3)	at 2
clear (?block)	at 2	on (block3, block1)	at 2

All these can be supported because they are, in fact, asserted in state 1. For each a constraint would need to be inserted from state 1 to state 2 to ensure that the support is not removed. In the case of two of the facts the place holder '?block' must be bound; 'block (?block)' can be supported either by binding '?block' to 'block1' or 'block2'; 'clear (?block)' however may only be supported by binding to 'block2'.

§5.2.3.2 Eliminating Conflict

In order for 2 states to conflict the following temporal condition must hold

> The first starts before the second ends and ends after the second starts.

In addition the two states must assign different values to the same fact.

If we, for the moment, ignore the complexities of the representation and just think about how the states might overlap in time then we can get four possible situations which we might show diagrammatically as shown in figure 5–31, where the horizontal extent of the shaded boxes represents the time period over which the state is assumed to hold.

Since any manipulation which we perform must be such that it reduces the amount of conflict in the plan rather than increasing it, it is a necessary condition that any

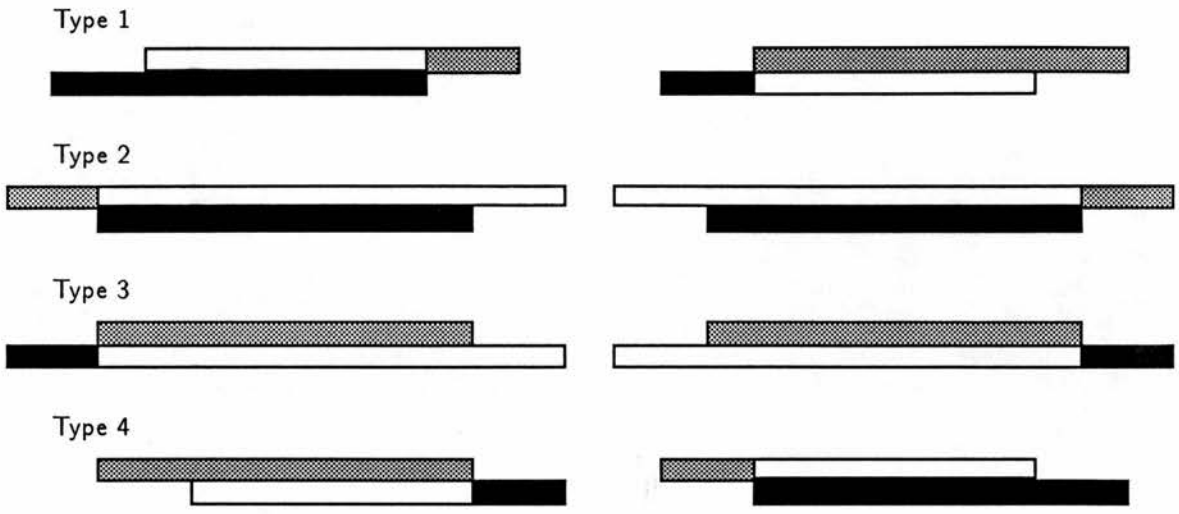


Figure 5–32: Solutions To the Different kinds of Conflict

manipulation of the start and end points of a state must be to restrict the period over which it is assumed to hold. That is the start time may not be moved back to an earlier time and the end time may not be moved forward to a later time because doing either might cause other conflicts to be introduced to the plan. Indeed since states are introduced with their extent set to be the maximum possible, if there is any position to which the end could be moved it must cause a conflict because only such a conflict could have caused the states extent to have been restricted previously. This is similar to the clipping done in the representations described in [Dean and McDermott 1987] and [Steel and Leung 1989].

This restriction on the movement of the end points of states means that only a few of the possible changes in the relative positions of states in a plan are legal. The changes which will eliminate conflicts of the four types in figure 5–31 are shown in figure 5–32, the white rectangle in each case indicating the original span of the newly restricted state.

With these examples in mind we can formulate the rules which will describe the legal and effective manipulations to remove conflict from situation in which one state conflicts with another, for convenience call the two states 'ϕ' and 'θ'.

> If the start of state ϕ is earlier than the start of state θ then moving the end of state ϕ to coincide with the start of state θ will result in a legal, conflict free state.

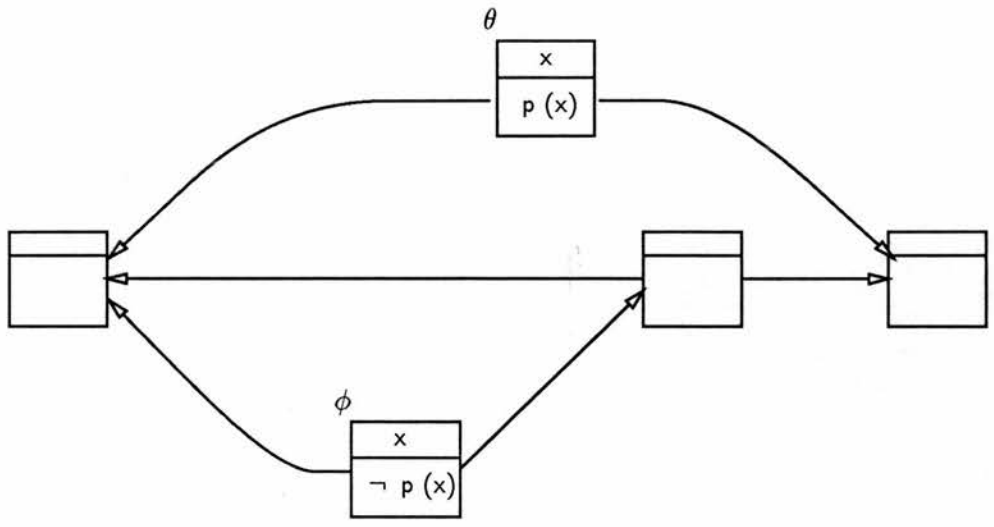


Figure 5–33: A Plan Fragment containing a Conflict

If the end of state ϕ is later than the end of state θ then moving the start of state ϕ to coincide with the end of state θ will result in a legal, conflict free state.

As shown above these rules will give one or two fixes for each conflict. In fact since the problem is symmetrical there will be two or four solutions, depending on which state is taken as 'ϕ' and which as 'θ', however the two pairs of solutions will have identical effects.

Now we have determined the abstract effects which we need to achieve we must apply this to the actual representation to be used by the system. Moving the beginnings and ends of states is precisely what the representation was designed to facilitate, so the correct link can just be moved. However there will be a problem when one or both of the links of a particular state are fixed. As an example take the plan fragment shown in figure 5–33. Here the conflict between the states marked 'ϕ' and 'θ' satisfies the first of the rules given above and so the correct fix is to move the finish point of state 'ϕ' back to state 'θ', as has been done in figure 5–34.

However if state 'ϕ' had happened to have had fixed limits this could not have been done; this is precisely what fixed links were introduced for. This case is shown in figure 5–35. In this case there is still a possible fix (there is, of course the mirror image fix which involves moving the end point of state 'θ'. However we are for the moment concerned with just the manipulations of 'ϕ' which will

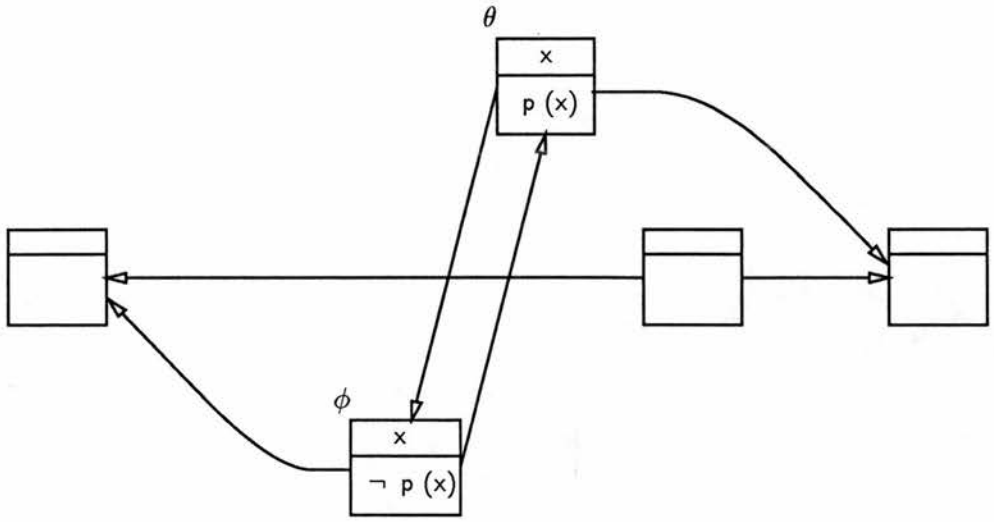


Figure 5–34: After eliminating the conflict

eliminate the conflict). We can recursively apply the fix which was meant for 'ϕ' to the state to which the endpoint which needs to be moved is fixed, here labelled 'ψ' The resulting plan fragment is given in figure 5–36.

As a final example figure 5–37 is a case in which there is no fix related to 'ϕ'. Intuitively, here we have a situation in which the states 'θ'and 'ψ' are seen to be ordered and 'ϕ' is constrained to begin as early as 'θ' and last until 'ψ' becomes true, there is no way of removing such a conflict by restricting 'ϕ'. Applying the

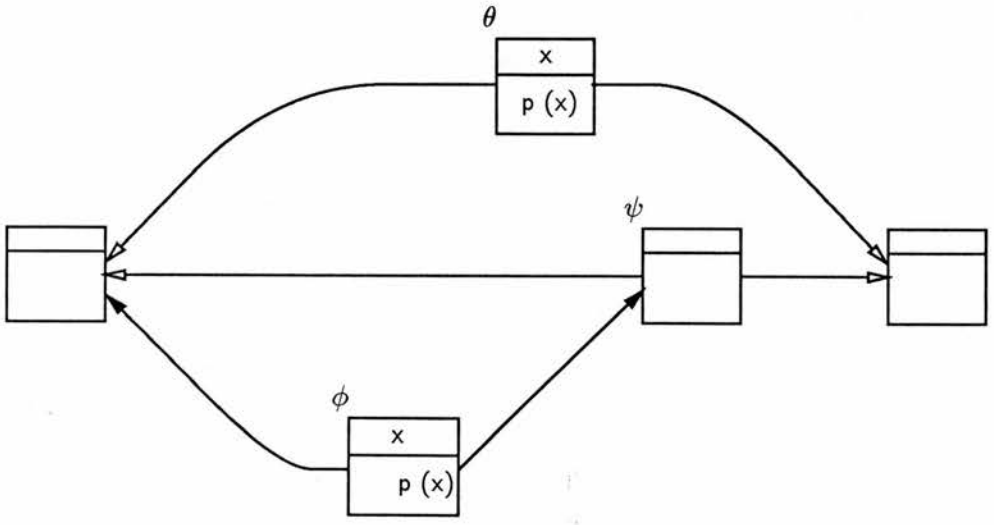


Figure 5–35: A Conflict involving a fixed link

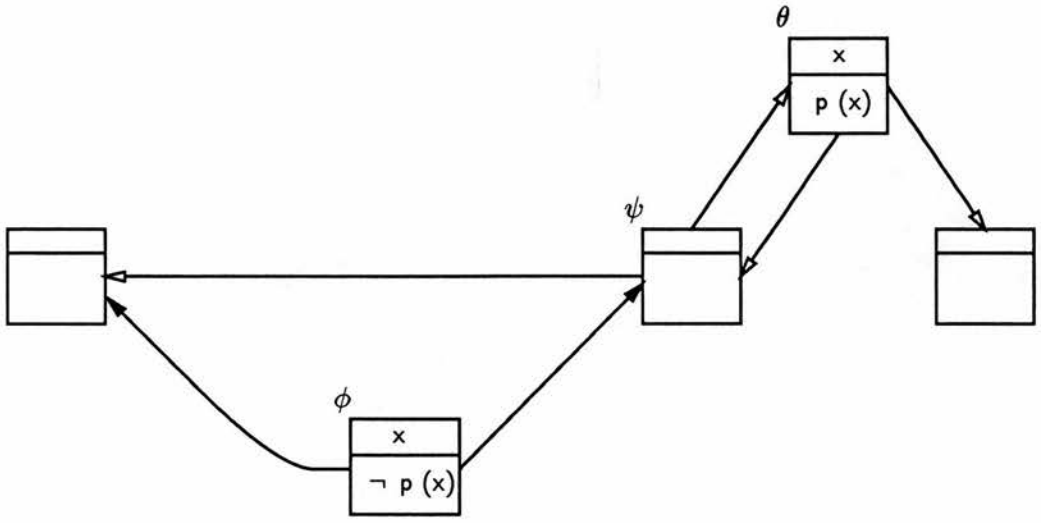


Figure 5–36: Eliminating the conflict involving a fixed link

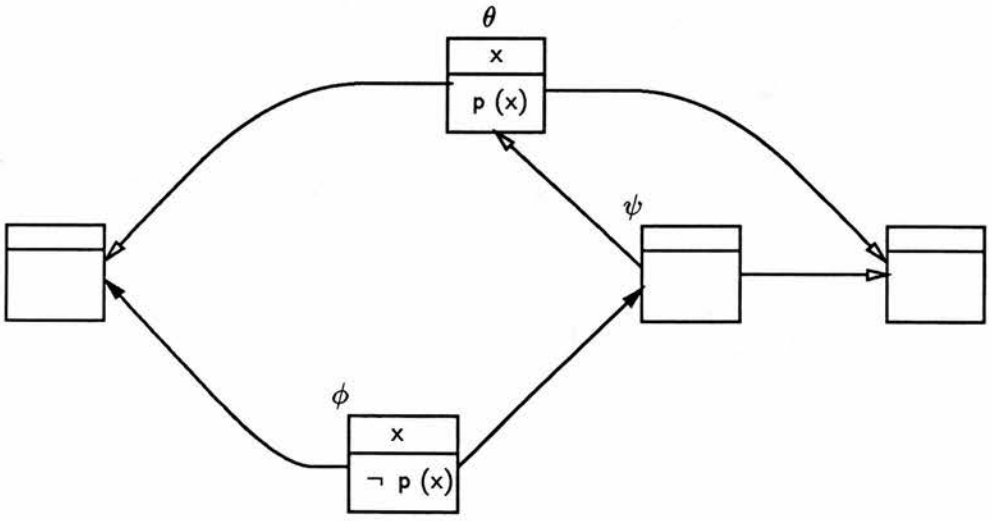


Figure 5–37: A Conflict Where ϕ cannot be restricted

rules which we have so far given we see that they do, in fact, predict this; 'ϕ' has fixed links and so we apply the fix to 'ψ' where its link is attached; 'ψ' however cannot be moved back to before 'θ' because that would put its ending time before its start!

§5.2.3.3 Binding a Place Holder

When the system needs to bind a place holder there are two options

An Existing Entity The simplest option is to bind the place holder to an entity which is known to exist. This will mean that the place holder has served as a synonym for that entity during planning. An example of this is '?block' in the example plan fragment; this place holder simply stands in for some place to put the moved block which we specify at a later time.

A New Entity The system may choose to introduce a new entity into the domain and bind the place holder to that. This is the case when, for instance, the system plans to make an utterance and associates various kinds of information with this proposed utterance in its planning. When the system comes to make the utterance, the place holder for the new entity must be introduced and bound to the place holder to represent the new utterance.

This latter case is a quite general method of introducing new entities into the universe of the system. For instance if someone was to open a conversation with the system by saying 16.

16) I met a police officer yesterday.

Then the system, when interpreting this, would have to leave a place holder in its universe, and plan, to stand for the policeman since the speaker might later say either 17a or 17b.

17a) It was Kate.

17b) I'd never seen him before.

In the first case the system would need to transfer any information which it had gained about the police officer to be about Kate. In the second it needs to introduce a totally new entity into its universe to stand for this new policeman.

The decision of whether to introduce a new entity or not is, in general, very hard. However for our purposes we can specify that any entity which is mentioned in the post-condition of an action but not in the pre-condition can be assumed to be a new entity introduced by the action. The primary example of an action which introduces an entity in the domain of text planning is the action of making an utterance, it seems reasonable to say that the utterance does not exist before the action is performed. [Chapman 1987] mentions this form of entity introduction in passing and points out that it makes planning from even simple initial states potentially non-terminating. However here we are describing a system whose intended purpose is to model relatively open ended behaviour and so assurance of termination is not possible.

Binding a place holder is the most widely acting of the fixes available to the system since it might cause the meaning of many states and actions in the plan to change; the next time the system looks to find what new chores it must add to its agenda it must look at all the states which referred to the place holder since any of them might contain facts which must now be supported.

§5.2.3.4 Making a state 'Current'

A plan represents a projected course of events. The system builds a plan in order to be able to select actions to perform and an ordering for them which will satisfy its overall goals. This means that over the time period represented by the plan each of the states described in it must at some time represent the actual state of the world. To reflect this the system must incorporate each state into its model of the current state of the world as it becomes the beginning of the plan.

When a state fulfills the following four conditions:

Position The state's earliest time is the current state of the world.

Support All of the facts in the state are supported.

Consistency The state is involved in no conflicts.

Grounding All place holders in the state have been bound.

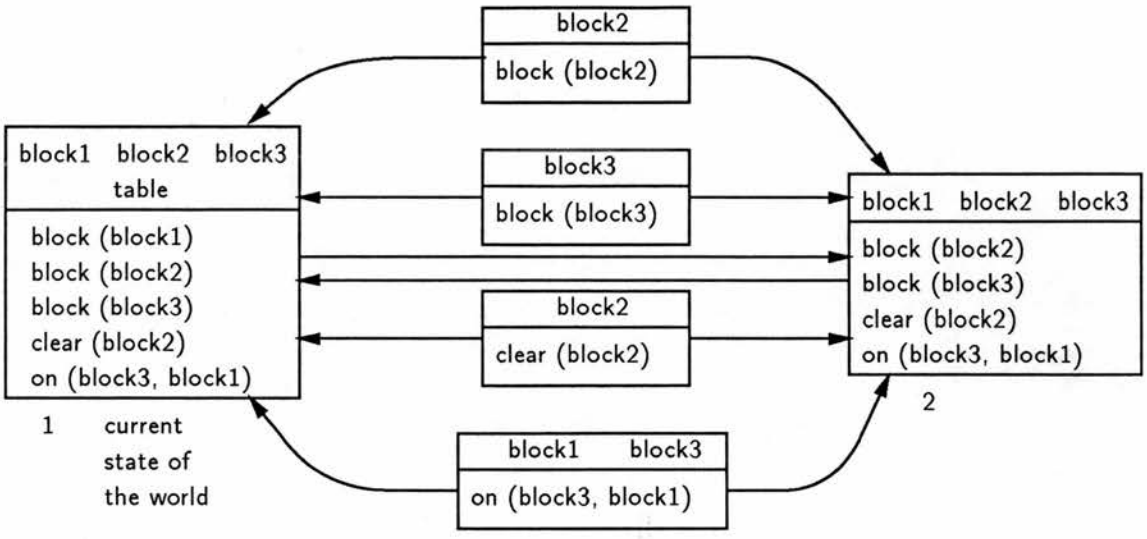


Figure 5–38: Part of the Example Plan After Support Has Been Provided

it must be true that the state describes the world as it currently is (although, naturally, it will probably not describe the current state of the world totally; it will describe some subset of it).

In such a case the system can merge the state into the current state of affairs with no loss of information. The procedure is simply to delete the state from the plan and make all the 'earliest' and 'latest' links which referred to it refer to the current state of the world.

As an example, assume that the place holder '?block' in the example plan fragment has been bound to 'block2' and that the facts in context 2 have been supported as described in section §5.2.3.1 . Here the four single fact states have been inserted as constraints to guarantee the support of the facts in state two. Each of these states along with state 2 satisfy the constraints on the 'make current' manipulation and so can be merged. The most complex case would be state 2 and so we will use that as the example. When state 2 is merged into the current state of the world, all links which refer to it must be transferred to the current state. The result, for the whole of the fragment not just the section shown in figure 5–38 is shown in figure 5–39. To simplify the diagram, we have included only one of the four 'constraint' states which were inserted between states 1 and 2; the other three would be in similar positions, with both beginning and end links connected to the current state of the world, and so still satisfying the conditions to be 'made current' in the future.

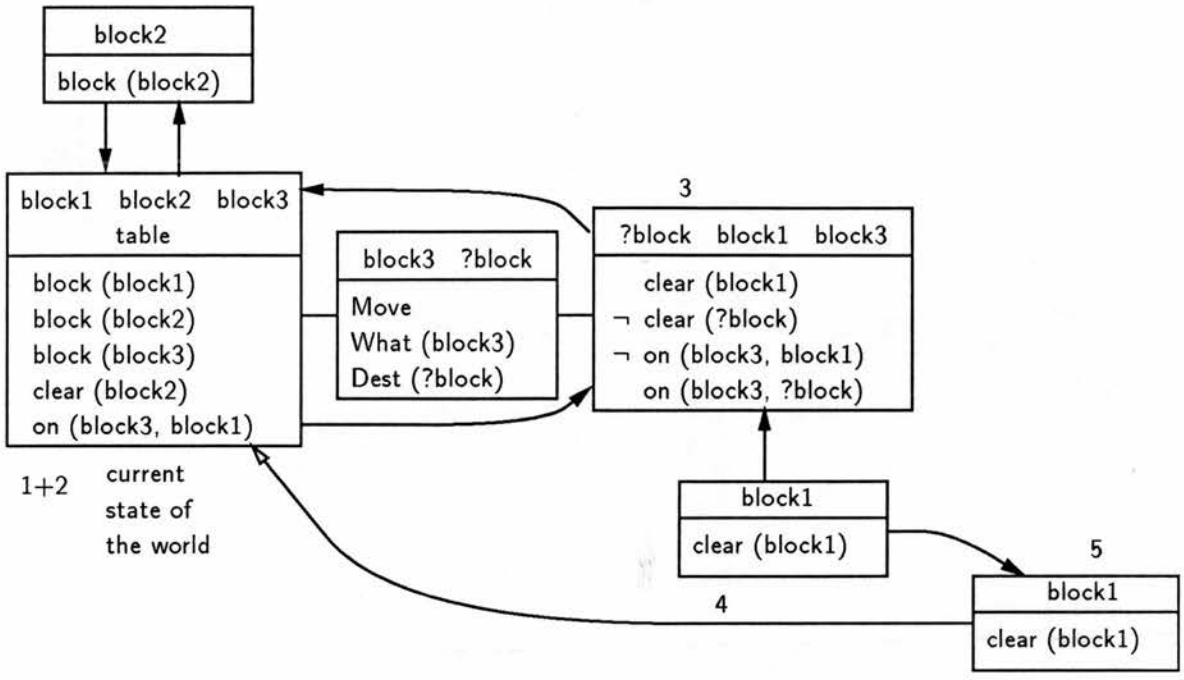


Figure 5–39: The Example Plan After Making State 2 Current

§5.2.3.5 Executing an Action

An action may only be executed if its pre-condition state is the current state of the world in the plan. Before the action is executed the system must remove the action and its post-condition from the plan and rearrange the links in the plan to reflect the removal. The rearrangement simply consists of making all links which referred to the post-condition now point to the current state of the world.

To actually execute the action the system must do one of two things

Symbolic Execution For simplicity we can assume that the action will have exactly the effects described in its post-condition, in this case we can just make those changes in the current state of the world. Presumably the system would also have to print a message or store a reminder to show that the action has been performed at this time.

Real Execution For a more realistic system we need only associate with each type of action a procedure to be executed to perform it. Such a procedure would take arguments corresponding to the place holders in the action's description. The procedure

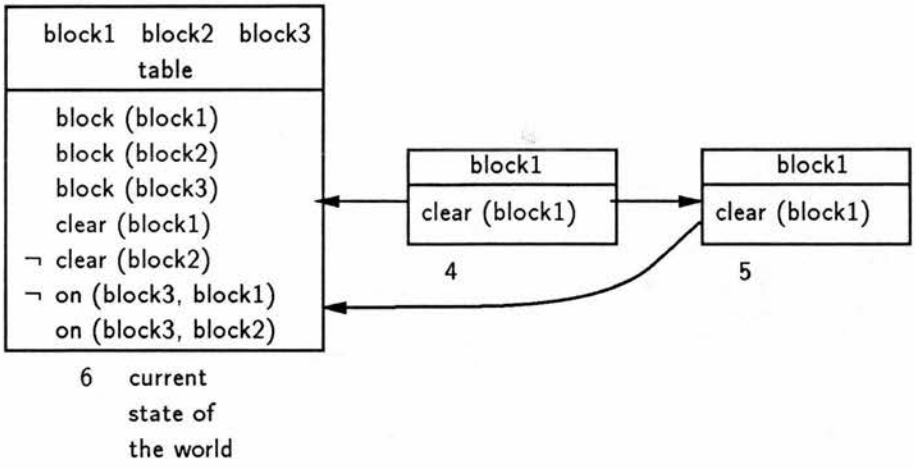


Figure 5–40: The Plan After Executing An Action

would have to return the actual changes which take place in the world and these would be incorporated into the current state of the world.

The actual implemented system provides for both of these options (by giving a default procedure which can be associated with an action to get symbolic execution).

Before the action in the example could be executed the place holder '**?block**' would have to be bound to 'block2' and state 2 would have to be made current as described in section §5.2.3.4. After this was done, if it were to be executed, the plan in figure 5–40 would result (we assume here that either symbolic execution was performed or the procedure associated with the 'move' action returned an indication that the result of the execution was exactly as expected).

One last thing to note about execution. Since it is impossible, in general, to undo an action in the real world the system cannot revert to an earlier state after an execution. This means that the 'backtracking' performed by the system must be limited to the time period between executions.

§5.3 Limitations and Extensions

The basic system described in sections §5.1 and §5.2 does not have all of the abilities which are required in order to work in a domain which involves communication. In this section we will explore the limitations of the system and introduce

extensions to overcome these limitations. The extensions introduced here are not as general as the basic planning constructs described in the previous sections, since some of the problems addressed are major research problems in their own right. However we have tried to produce extensions which make minimal assumptions about what full solution to these problems might be discovered in the future, in keeping with the 'Generality' constraint outlined in the introduction.

§5.3.1 Generating Goals

The system as outlined so far has no way of introducing new goals into its plan. All the goals which might be introduced in the described system must be the result of introducing actions into the plan which in turn must be the result of existing goals. The system needs some way to introduce new goals based on its planning.

Some of the situations in which a language using system might wish to introduce totally new goals are as follows —

Response to Others When something happens in the real world the system might need to introduce new goals into its planned behaviour so that it can take account of it. For instance when someone says (or types) something to the system it must respond in some way; a way to do this might be to introduce a goal to interpret the input. Similarly when an agent interpreting a text decides that the current utterance signals the start of a new discourse context, they must remember that this context must be closed again. This kind of goal adoption forms the backbone of the implementation of the 'points' interpretation model of chapter 2 which we shall use as an audience model in chapter 7.

Consistency Upon introducing certain types of assumption into its model of the world, the system might wish to note to perform some actions at a later time to verify the consistency of the new assumption. For example if the system decides to introduce a universal rule like "All birds fly south in autumn" into its beliefs then it might wish to check for every bird it knows if it is true.

Assuming Goals When we come to discuss planning for other agents to perform actions (in section §5.4.5) we will need to be able to induce other agents to assume some goal. Since all of the agents in a domain are assumed to be of the same kind, this means that we must have a way of assuming goals suggested by others.

The common factor in all these situations is that the system detects some set of circumstances in the world (or in its plan for the world) and introduces a goal in response. To capture this we extend the knowledge of the planner to include information about which states of affairs should trigger new goals and what goals should result. This information will be contained in 'rules' which consist of two parts —

Condition A pattern representing a class of situations (states) which can induce goals.

Goals A state which is to be introduced into the plan in cases where the condition occurs.

Such a rule can be thought of as a constraint on the set of legal plans – every plan which contains a state which matches the 'condition' of the rule must also contain (at a later point in the plan) the relevant instance of the 'goals'.

Since the system already has the ability to represent classes of states as states some of whose entities are treated as place holders, all that is necessary to add such rules to the system is a more precise idea of what we mean when we say a state 'matches' a pattern.

Given two state descriptions, both of which might contain place holders (since the plan may contain unbound place holders representing choices which have been delayed) we might apply a standard (graph-)unification algorithm to find interpretations for the place holders. In general there will be a number of such unifiers and each will represent an interpretation of the place holders in the rule and the plan which will cause the rule to fire. However this would cause our search for rules to prematurely make choices about the bindings of place holders in the plan and that is not what we want — the generation of goals was *meant* to be driven by the structure of the plan, not *vice-versa.*

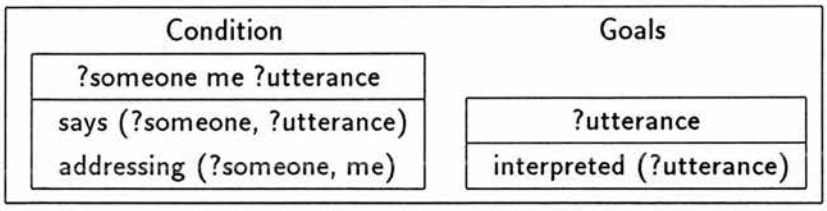


Figure 5–41: An Example Rule

fred me bill utterance12 utterance13
addressing (fred, me)
addressing (bill, me)
says (utterance12, bill)
says (utterance13, Fred)

Figure 5–42: A State Which Matches the example Rule

Instead we must treat the place holders in the plan as fixed entities when looking at rules to fire. There will still sometimes be more than one unifier, but each of these will represent a truly independent instance of the goal generating state of affairs.

An example of a rule of this kind is figure 5–41 which might be rendered in English as —

> Whenever someone who is addressing me says something then I must interpret that utterance.

If the condition of this rule is matched against the state in figure 5–42, where two people are addressing the system. The following two unifiers are produced

?someone = fred	?someone = bill
?utterance = utterance13	?utterance = utterance12

which, when applied to the goals part of the rule would produce the two goal states in figure 5–43. Given a set of such rules the system can determine what

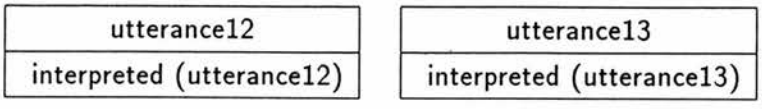


Figure 5–43: The Two States Triggered by the Example State

goals must be introduced, but it does not know where in the plan it should go. There are a number of possibilities —

Immediately The goal might be placed immediately after the state which matched the condition when the rule was triggered to be achieved and then forgotten.

Eventually The goal might be placed at the end of the plan, indicating that the required state should be made true some time in the course of the plan and remain true until the end of the plan.

Sometime The goals might be left to float between the triggering state and the end of the plan, to be achieved some time then forgotten.

In addition the above strategies could be performed with arbitrary points in the plan rather than the triggering state and the end of the plan; these might be specified in the rule as patterns to be matched.

For simplicity we choose to implement only the simple 'Eventually' case, since this allows for all of the behaviour which we wish to capture. For instance, in the 'points' interpretation model introduced in chapter 2, it is necessary to close a context before its parent is closed, but we do not need to encode this explicitly in the rule which asserts the goal of having the context closed, rather the actions which close contexts are such that closures must be correctly nested.

The modifications to the basic system to allow for rules are as follows —

- A set of 'rules' is included in the planners knowledge.
- A new type of chore is defined which indicates that a rule must be fired. When looking for chores the system must compare each of the changed states with each of its rules and if it matches add the chore to fire the rule.
- A fix is added for the 'fire rule' chore which inserts the correct instance of the rule's goals into the plan with its 'earliest time' link fixed to the end of plan state and its 'latest time' link at the state which triggered the rule, but able to be moved.

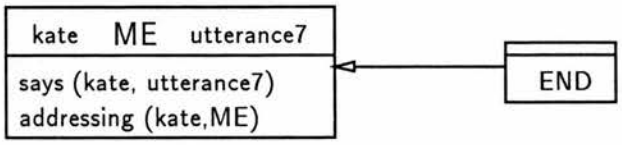


Figure 5–44: A Plan Fragment Which Will Trigger a Rule

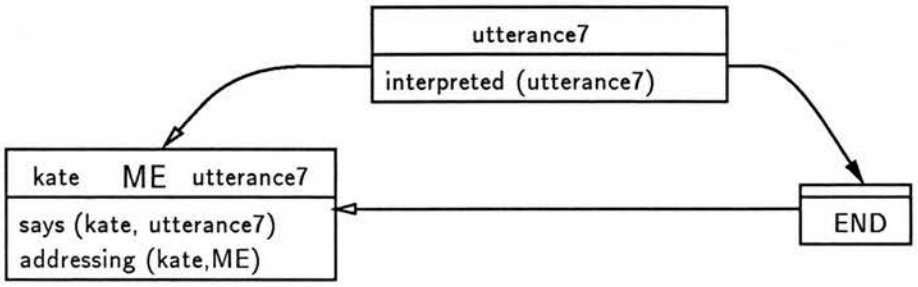


Figure 5–45: After Firing The Rule

Given these changes and the above example rule the system confronted with the plan in figure 5–44 would add a chore to fire the rule with '?someone' bound to 'kate' and '?utterance' bound to 'utterance7'. When it came to apply its fix for this chore it would get the plan in figure 5–45.

§5.3.2 Modal Facts

No mention has yet been made of how the planner is to deal with what were called 'modal' facts in section §4.3. If we were to treat them just like the non-modal facts we have been using so far there would be a number of serious problems.

- Almost any two states which give values to the same modal fact and happen to overlap would conflict. It is very unlikely that two states would ascribe precisely the same beliefs to an agent, for example, though the beliefs ascribed should be compatible.

- Conversely it would be almost impossible to find support for such a fact since there is unlikely to be either a state in the current plan or an action which gives the fact the required value.

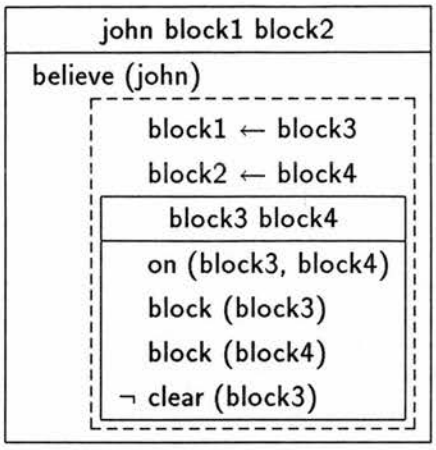


Figure 5–46: A State Containing A Modal Fact

- Rules would be similarly affected since each rule whose condition contains a modal fact would need to take into account every possible value which the fact might have and still trigger the rule.

To avoid these problems the system must treat the values of modal facts as something more finely distinguished than the True/False for normal facts. The most intuitive way of doing this is to divide a modal term–value pair up into some kind of atomic units. The obvious 'unit', especially since we are interested in linguistic actions, is something which might be expressed in a single (modally qualified) clause. In this sense, figure 5–46 asserts four facts and not just one; these facts might be rendered into English as

John believes block two is on block one.
John believes block one is a block.
John believes block two is a block.
John believes block one is not clear.

To treat modal facts in this way (which might be called 'distributively') we must introduce the concept of a (non-modal) fact combined with a sequence of zero or more modal-facts indicating how it is to be understood; we call such a combination a "complex fact". The above English sentences might be represented by the complex facts

believes (john): on (block1, block2)
believes (john): block (block1)
believes (john): block (block2)
believes (john): ¬ clear (block1)

And a more deeply nested complex fact might be

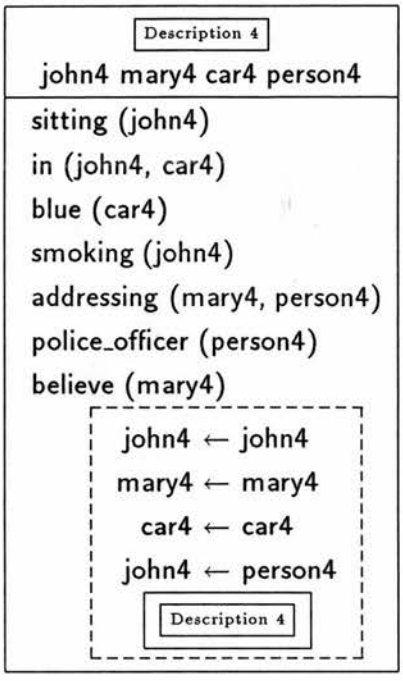


Figure 5–47: A Description With Mutual Belief

believes (john), believes (mary), wants (kate): on (block3, block1)

Meaning something like:

> John believes that Mary believes that Kate wants block one to be on block three.

With this concept we can describe how the system should treat modal facts. When looking for conflicts and for support the system should look at the complex facts in each state in the plan not at the facts. When matching a rule against a state in the plan it should match complex facts and when looking for an action to provide support it should check the complex facts in the post-condition.

There is, however, a problem with this concept of "complex facts"; when looking at a view which contains circular references it is difficult to see when to stop looking for new deeper complex facts. For instance, take the example given in section §4.3 of a circular belief structure, repeated here as figure 5–47 for ease of reference. Here the beliefs ascribed to the entity 'mary4' (from here on just mary) are this very same state with a different interpretation given to the entities. The actual complex facts here are

sitting (john4)

in (john4, car4)
blue (car4)
smoking (john4)
addressing (mary4, person4)
police-officer (person4)
believes (mary4): sitting (john4)
believes (mary4): in (john4, car4)
believes (mary4): blue (car4)
believes (mary4): smoking (john4)
believes (mary4): addressing (mary4, john4)
believes (mary4): police-officer (john4)

However, a simple-minded system would either not include any of the complex facts about Mary's beliefs (since they are represented by the same state, it would detect a cycle and stop), or would continue inserting

believes (mary4), believes (mary4): sitting (john4)
believes (mary4), believes (mary4): in (john4, car4)

and so on, since the interpretation of the entities changes each time around the cycle. Because of this type of example (which is going to be common whenever we are dealing with mutual beliefs) we must take care in the definition of what complex facts are asserted in a state.

We can think of a (possibly cyclic) system of views containing other views as a shorthand for a (possibly infinite) tree of simple views where each branch is a path which might be taken from the outermost view using the rules of figure 5–15 to combine the mappings of each modal fact we pass through into a complete view of the place we are at as seen from the root. This way of dealing with cyclic structures has the advantage of simplicity and allows a set of nested views to be mapped to a simple non-cyclic interpretation which speeds up the common operations of matching and conflict detection. The major disadvantages of this approach are that it hides the *structure* of the states being examined — we can not represent concepts such as "States in which some Kate is mutually believed by the speaker and audience to be a police officer" in a way which is useful — and that it can sometimes miss possible matches or conflicts between cyclic structures which would be caught by a system which employed a cyclic structure unification algorithm on pairs of states. We accepted these restriction in return for the performance advantages because they do not prevent the system from doing interesting text planning and because we believe that the text interpretation and planning systems

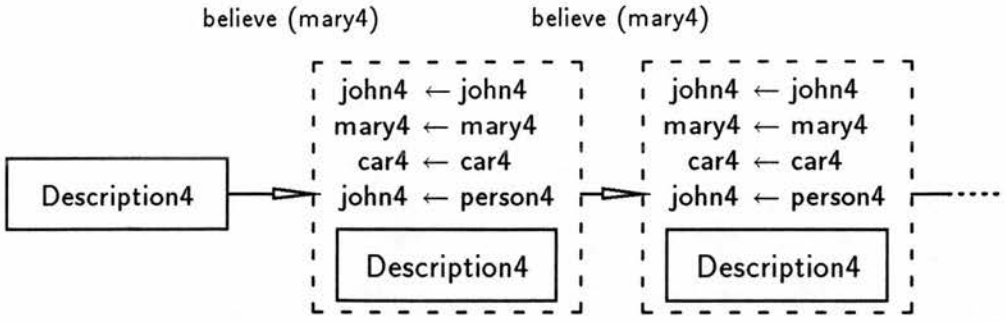


Figure 5–48: The Tree Structure Represented by Figure 5–46

we have created is not dependent on these restrictions but could be transferred to a more capable system if one was created.

The unwound tree structure for the view in figure 5–46 is shown in figure 5–48. In this simple case the tree consists of just one infinite branch containing a sequence of similar views.

Using this concept we can define the set of facts given values by a view 'V'to be the set of complex terms consisting of a sequence of modal facts representing a path through V's tree which does not contain two equal views, followed by any boolean fact from the final view in the path.

Intuitively, the way to avoid cycles is to remember the "state" of the search for interesting complex facts at every stage and stop if we ever attempt to return to an earlier "state". As indicated above the interesting information about a "state" will be not just the context which is reached but also the interpretation of entities in that context; thus the "state" will be a view.

At each step of the search the system must remember

path A complex (modal) term which is known to be a valid prefix of some of the complex (boolean) terms which the search must find.

old A set of views representing previous states of the search.

current A view which is the current search state.

The search proceeds by examining each term 't' which is assigned a value in 'current'. If 't' is modal then the search continues with 't' appended to 'path',

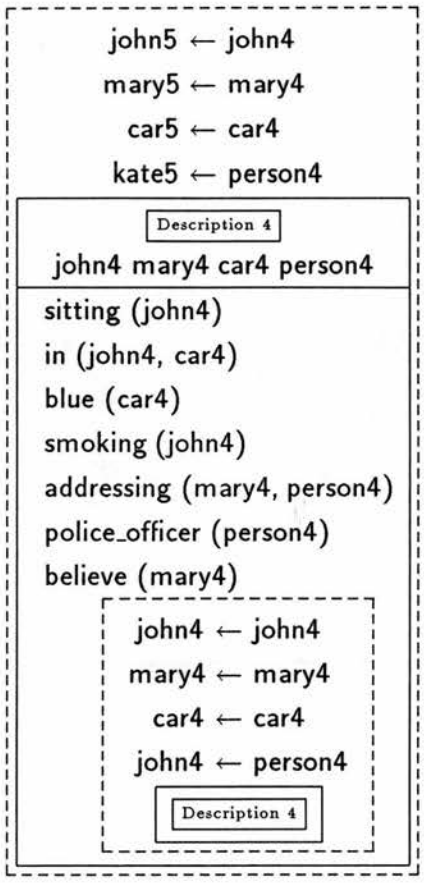


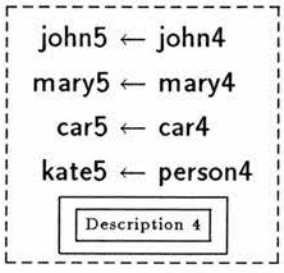
Figure 5–49: A View With Mutual Belief

'current' added to 'old' and the value of 't' in 'current' as the new 'current'. If 't' is boolean then the complex term formed by appending 't' to 'path' is a result of the search.

The search is started with 'path' and 'old' empty and 'current' the state in the plan whose effects are to be determined. If at any point 'old' contains 'current' then that branch of the search is abandoned and another is tried.

As an example of this process, consider the view in figure 5–49, constructed from the description of figure 5–47. The search will start with an empty path, an empty set of views and current equal to the view in figure 5–49.

Path = "" Old = {} Current =



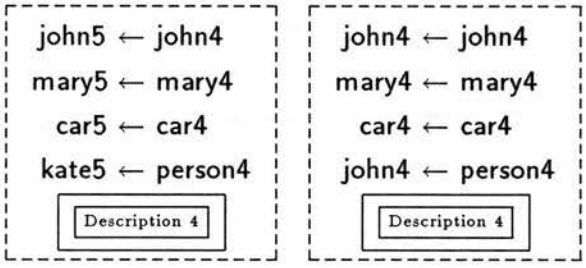
The boolean terms which are given values in 'current' are

sitting (john5)
in (john4, car5)
blue (car5)
smoking (john5)
addressing (mary5, kate5)
police-officer (kate5)

The only modal term which is given a value is

believe (mary5)

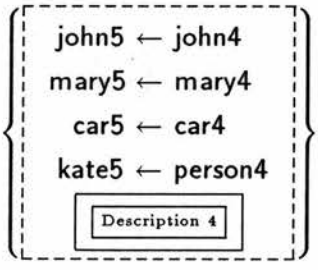
The value assigned to this modal term will be the combination of the two views



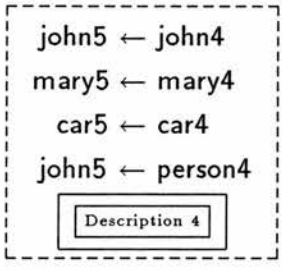
The next state in the search will be

Path = "believe (mary5)"

Old =



Current =



The complex terms which can be built from 'path' and the boolean terms in 'current' are –

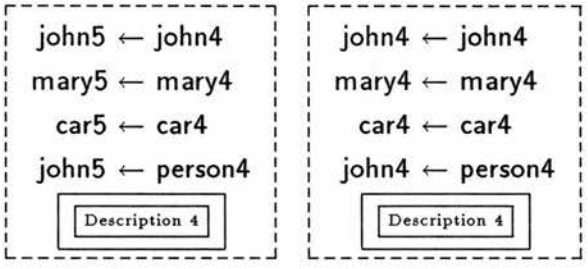
believe (mary5): sitting (john5)
believe (mary5): in (john4, car5)

believe (mary5): blue (car5)
believe (mary5): smoking (john5)
believe (mary5): addressing (mary5, john5)
believe (mary5): police-officer (john5)

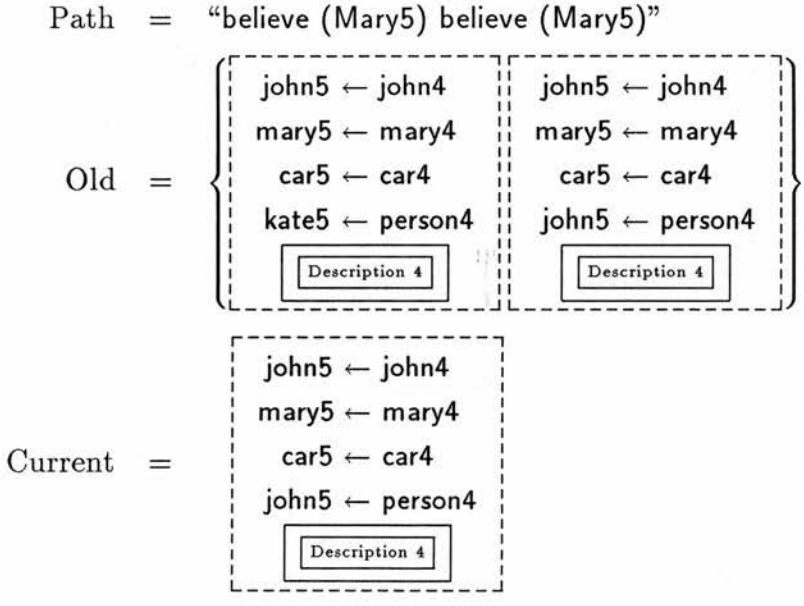
Once again there is only one modal term given a value –

believe (mary5)

The value assigned to this will be the combination of



So the new state will be



But here 'current' is a member of 'old' and so the search ends for this path. Since there were no choices of which path to follow, the search terminates and all complex terms given a value by the view of figure 5–49 have been found.

§5.3.3 Quantification

In general the problem of introducing quantification into the representation used in a planning system is very complex. A balance must be maintained between

the power of the representation and the added complexity it introduces into the planning process. For the purposes of simple text planning, we can limit quantification to quite simple cases which are needed to implement the kind of interpretation model described in section §6 of chapter 2 (an example of a system which has tackled this problem in a more general manner is SIPE described in [Wilkins 1988]).

To implement even simple interpretation processes we will need to introduce quantification into the planning process in two places —

- When describing the beliefs of agents about the 'meanings' of utterances of various forms we want to be able to say "all utterances with an agent and a patient and the predicate 'owns' have meanings of the form '**owns (?agent, ?patient)**' ".

- When describing inference rules which an agent may apply in interpreting a discourse we may need quantification. In this work we will only be interested in inference rules of the form "in all cases where X holds, Y holds".

Both of these kinds of knowledge are simple forms of universal quantification and fortunately we can express this kind of quantification in our representation very simply and interpreting it as part of the planning process is also quite straightforward.

We can represent this simple form of quantification by creating views which do not map all of the entities in their context onto entities in their surroundings. Thus we might use a view of the form

as a description of a situation where **a1** is treated normally but **b1** and **c1** are treated as universally quantified. In doing this we are exploiting some of the redundancy of the views representation in perhaps a non-intuitive way. A more direct interpretation of a view which does not supply interpretations for all of the entities in its context is that it gives values only to those terms given values in the context which contain only interpreted entities. However, since we have no use

for such views, it seems reasonable to use them to represent quantification rather than to introduce additional structures to do so.

This is simply the form of quantification which we might get in a logic by stating that free variables in a formula are implicitly taken to be universally quantified and disallowing explicit quantification.

Planning with this kind of representation is also fairly simple. We must simply make sure that the unification process which is used to match facts to be supported or rules which may be triggered does not attempt to bind such 'dangling' entities to entities in outer contexts. This restriction will allow a complex fact of the form

```
foo (), [ ?x, ?y, ?z ] p (?x, ?y, ?z)
```

where the square brackets are used to represent quantification, to support any number of unsupported complex facts of the form

```
foo (), p (a, b, c)
```

since the place holders ?x, ?y and ?z will not be bound by each support fix.

§5.4 Other Agents

The most striking way in which the system so far described is insufficient for chores involving communication is its total lack of any facilities for taking into account the goals, abilities and actions of other agents. In this section we will develop a simple model of interaction with other agents within the assumptions described in section §5 of chapter 3.

The most important assumption which is made by this system is that called 'agent similarity' in section §2; all agents who are active in a domain are assumed to be of the same type, they are assumed to be systems of the type being described here.

Another important assumption is that it is unnecessary to model the details of the planning activities of other agents; it is assumed that it is enough to take into account that an agent will be performing a certain action within some span of time and the system need not model the precise reasoning which it believes that agent to have followed in their decision to perform that action. What it *must* know is the *order* in which the other agents will examine actions in an attempt to find

one to support a goal. It must know this because it needs to be able to rule out the possible supporting actions which would be examined before the one which it wishes the agent to use. The assumption that the detailed mechanisms of an agent's planning need not be modelled means that an agent's state at some time can be represented by their beliefs, without any notion of their planning activity.

Given these assumptions there are 6 parts to the modelling of other agents which the system must perform

Beliefs The system must have a consistent way of representing the beliefs of other agents.

Plans Other agent's plans must be represented within the system in such a way as to indicate their relationship to the system's own plans

Goal Adoption The adoption of goals by agents is one of the most important parts of interaction, so this must be modelled in some way which is consistent with the rest of the system and relatively efficient.

Observing Actions The system and other agents must be able to see when other agents perform actions and note their effects.

Action Planning The system must be able to reason about how actions by other agents are going to affect its own plans. Actions by other agents must be represented in the plan of the system in such a way as to make it clear who is to perform the action and what effect the system expects this action to have — it is not enough to represent the effects which the agent performing the action believes it will have since that agent's beliefs about the preconditions of the action may be quite different from those of the system.

Action Execution Actions which are performed by other agents must be handled in a different manner to the system's own planned actions when they become eligible for execution. In the simplest case the system must be able to wait for the other agent to perform their action (for instance waiting for another speaker to finish an utterance before taking a turn).

The following subsections discuss these facets of interaction with other agents.

§5.4.1 Other Agent's Beliefs

The basic system already has the ability to represent multiple models of the world as views which are the values of modal facts, so to represent the beliefs of other agents it is only necessary to define a modal functor 'believe' of arity 1 which can be used to describe the beliefs of other agents. The value of a modal fact of the form

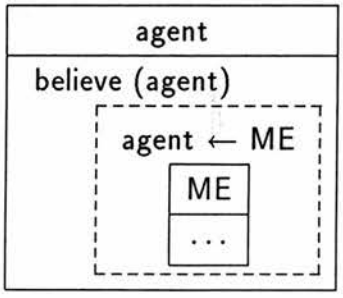
believe (agent)

will be a view describing the beliefs of agent.

However there is more to representing the beliefs of agents than this. There are a number of properties which we require to be true of the beliefs of any agent —

- Agents know of themselves; each belief set must contain an entity, which we will in the following discussion write as 'ME', which represents the agent's concept of themselves.

- Agents know about other agents' concepts of themselves. This self concept 'ME' will be distinguished to the agent manipulating the belief representation by the fact that it is the entity which is mapped by the view to the argument of the believe term. That is, a believes term and its value will always be of the form



- Agents believe their own beliefs and believe that they believe them. The value of the term

  Believe (ME)

  in a description of an agent's beliefs will be that set of beliefs, thus the following are all equivalent complex facts

violinist (kate)
believe (ME) : violinist (kate)
believe (ME), believe (ME) : violinist (kate)
⋮

and so on.

Notice that the second and third property will together ensure that the following complex facts are all equivalent

believe (mary) : violinist (kate)
believe (mary), believe (mary) : violinist (kate)
⋮

In order to ensure that these properties always hold, we place restrictions on the configurations of views which can form states in plans. To describe these restrictions it is useful to define the concept of a "belief view" as follows —

- A 'root' view which is used to describe a state in the plan is a belief view for the agent 'self' which exists in any planning domain to represent the planning system itself.

- Any view which is the value of a term 'believe (*agent*)' *in a belief view* for some agent is a belief view for *agent.*

The restriction we place on configurations of views used to represent states in the plan are —

- Every belief view for an agent *agent* must contain an entity '*ME*' which the mapping of the view maps to *agent.*

- In any belief view the value of 'believe (ME)' is a view with the same description as the belief view and a identity mapping (i.e. one which maps all entities to themselves.

Ensuring that these properties hold does not involve as much work as might be imagined since these conditions need be enforced only once, when the definitions of actions and rules which are to be used in constructing the plans are read in. Only the one-of 'constraint' states introduced in providing support (see section §5.2.3.1) need to be examined for belief views when they are inserted.

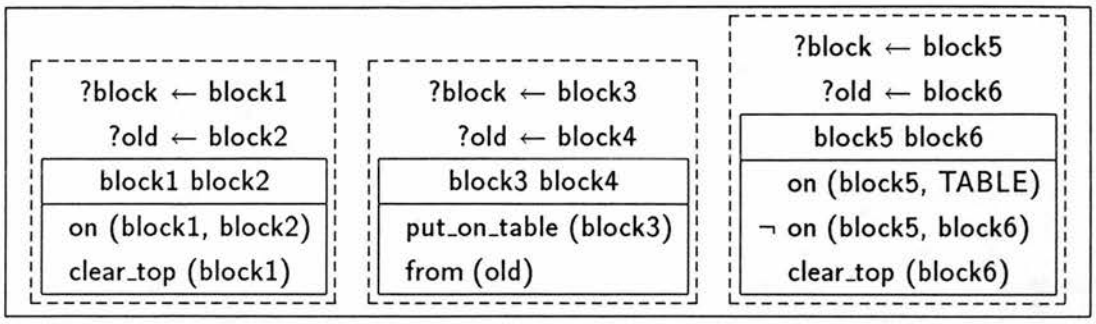


Figure 5–50: The Action put-on-table

§5.4.2 Other Agent's Plans

Since we assume that it is not necessary to represent the detailed state of an agent's planning at all points in the system's future plans, but only the actions planned, their expected results and their timing with respect to the system's own plans, all that is needed to represent multi-agent plans is a way of inserting other agent's actions in the system's plan.

However it is not sufficient to simply tag actions in some manner with an agent who is to perform it. An action which another agent is expected to perform may have preconditions which that agent believes satisfied but which the system believes not to be. In such a case the results of the action in the system's plan might be very different to those which its performing agent expects.

This means that the pre- and post-conditions of an action instance inserted into the plan should be not those defined in the action description but rather, should be constructed from the definition taking into account the identity of the agent performing the action and the planner's model of that agent's belief at the time the action will be performed.

As a simple example, consider the action '`put-on-table`' which moves a block from its current position onto the table in the blocks world domain. This action might be defined as shown in figure 5–50. Assume the system wishes to have the block 'big_blue_block' on the table. The simplest way to achieve this would be to plan to perform an instance of `put-on-table` with '?block' bound to 'big_blue_block'. If, however, the system could not perform this action, but the domain contained another agent who could (say 'Kate') then the system could plan to have that agent perform the action. If the system and Kate agree on the identity of big_blue_block then the operation is simple, the action could be inserted into the plan as normal and marked in some way as being performed by Kate. However this does not

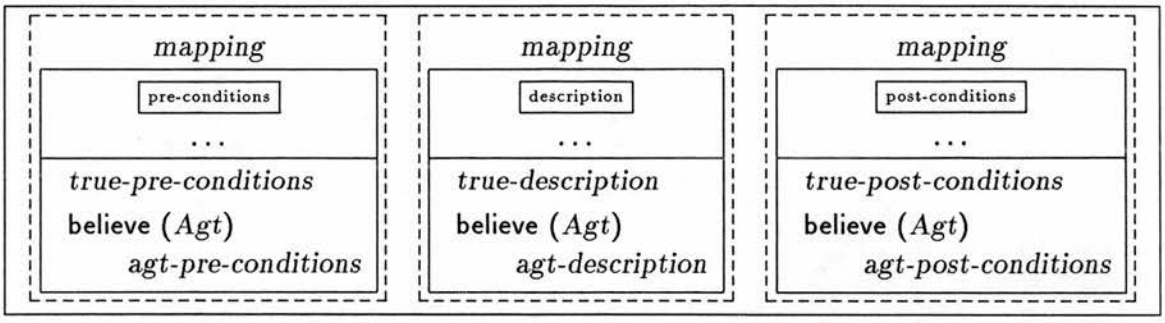


Figure 5–51: The Form of an Action by Another Agent

generalise to more complex cases and so we use a more complex, but more general, representation for actions by other agents.

For example, if Kate and the system disagree over the identity of big_blue_block then the planning process in the above example is more complex. Assume that Kate believes that the entity which the system knows as big_blue_block and the entity the system knows as large_block are the same entity. Then in the state of the world resulting from the `put-on-table` action both of the following complex terms must be assigned the value TRUE –

believe (Kate): on (big_blue_block, TABLE)
believe (Kate): on (large_block, TABLE)

A general method for dealing with this problem is to insert actions to be performed by agent 'Agt' into the plan with pre- and post-conditions and description of the form shown in figure 5–51. Here, 'pre-conditions', 'post-conditions' and 'description' are taken from the definition of the action while '*mapping*' is the binding of the action's place holders. The *true-pre-conditions* describe the state of affairs the system actually believes must hold for the action to be performable by the other agent and the *true-post-conditions* describe the actual expected results. The *agt-pre-conditions* and *agt-post-conditions* describe the action as it is presumed to be perceived by Agt. Both the actual expected preconditions and results and those to be perceived by the other agent must be included so that future planning can ensure both that the actual action is possible and that the other agent will believe it possible — otherwise they might perceive a conflict which they would attempt to avoid, so changing their behaviour in an unexpected way.

In the example of getting Kate to put big_blue_block on the table, the instance of `put-on-table` which must be planned is shown in figure 5–52 (only the before and after states are shown for reasons of space).

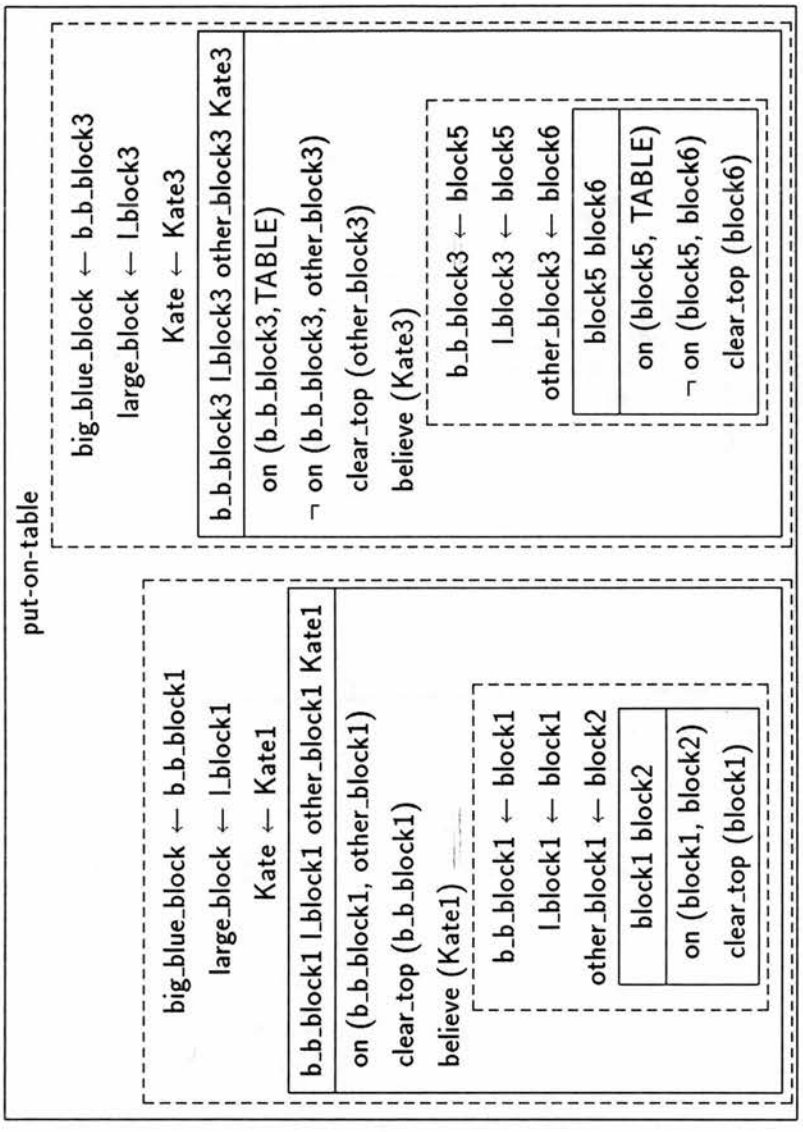


Figure 5–52: put-on-table to be Performed by Kate

One final thing needs to be taken into account when designing a representation for multi agent plans. Each agent can be assumed to have a model of the actions of each of the others, including those other agents' models of themselves and third parties. In a situation involving many agents a full representation could therefore be very complex. In fact it will rarely be the case that an agent will make detailed plans for the plans of all other agents; like human agents they will normally assume that the actions of other agents in the vicinity will not be of importance to their plans except when they know that some other agent will purposely be interacting with them. Thus the system need build only as many layers into its plan as it is forced to; when it determines that it must add an action by another agent into the plan then it will add just the information which this action involves (two states

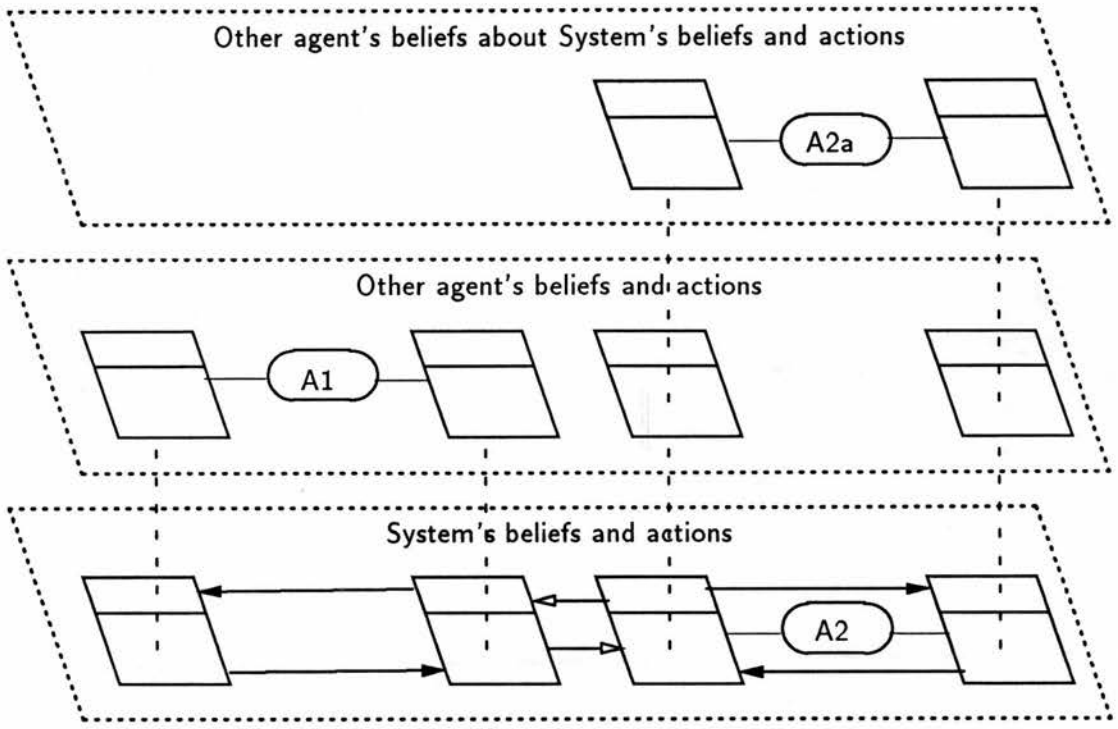


Figure 5–53: A Plan With Actions for Two Agents

and the action assuming the model as presented so far – section §5.4.4 discusses situations when more information must be inserted).

Given the above decisions, a plan in a domain with multiple agents can be visualised a stack of parallel plans, the lowest containing the actions of the system, the next higher levels containing beliefs and actions of other agents, the next higher levels containing other agents plans for the action of third parties and so on. For the simple two agent case (just the system and one other agent) the developed plan would have a structure something like figure 5–53. Here action 'A1' has been planned by the system to be performed by the other agent while 'A2' is a planned action by the system which is visible to the other agent and so is included in the other agent's beliefs about the system (see section §5.4.4 for more details of action visibility). The vertical lines represent the **believe** relations in the views representing states in the plan. Note that the parallel plains here are implicit in the representation rather than being explicitly present as shown.

§5.4.3 Goal Adoption and Support

Since all agents in a domain are assumed to be similar, each of them will be working under constraints in the form of goal generating rules as described in section §5.3.1. This means that the system in planning its course of action must take note of when any of the agents believes at some point that the condition part of a rule is satisfied and when this occurs it must add to its agenda the chore of finding support for the goals which will be introduced.

It is not however simply a case of inserting all these goals into the plan as was described in section §5.3.1; if a goal is introduced into the plan as a result of a rule matching the beliefs of some agent other than the system, it will be that agent's goal and so it will be that agent's responsibility to provide support for that goal. However since this system is meant to be 'subjective', that is taking the part of a single agent with no control of the other agents, it will be necessary for the system to make assumptions about the way in which the other agents will support their goals so as to take their actions into account in its planning.

There is therefore a need to extend the definition of the 'prove support' chore of the system to take into account a parameter indicating who is responsible for finding the support. To supply this parameter it would only be necessary to mark each belief view (as defined in section §5.4.1) in the plan with who is responsible for supporting the facts in that state. As was mentioned in section §5.2 of chapter 3 we adopt a relatively simple strategy for assigning responsibility. In fact we use just the following two rules —

- Belief views introduced as a result of a rule being matched by a belief view of an agent will be the responsibility of that agent.

- Belief views introduced as parts of actions which were inserted to provide support for a fact which is the responsibility of an agent will be the responsibility of that agent.

Notice that the second rule does not refer to which agent is performing the action, but rather to which agent's goals the action was introduced into the plan to serve. This means that if the system has a goal which it can not support by its own actions and so plans for another agent to perform an action, it will be the system's responsibility to support the preconditions of that action. On the other hand if a rule fires on a belief view of another agent as a side effect of the system's planning,

supporting the resulting unsupported complex terms will be the responsibility of that agent. Naturally the system must still construct the other agent's (presumed) plan since it will need to know how that plan will interact with its own plans.

Once it is known which agent is responsible for providing support for a given complex term, the support process proceeds from that agent's 'point of view'. That is, the fixes of section §5.2.3.1 are all interpreted with reference to the system's model of the responsible agent's beliefs. For instance, the fix 'Made True Earlier' should be read as

Made True Earlier If the complex fact is asserted earlier in the plan *as a belief of the responsible agent* then it can be supported by inserting a constraint which assures that the fact will remain asserted *in that agent's beliefs* between its earlier assertion and the state needing support.

Thus the major effect of 'responsibility' will be to prevent the system from planning actions to provide support of other agent's goals except where they directly aid the system's plans. This means that the system is not as helpful to other agents as it might be. However to make it more so we would need to define rules for when it is reasonable to be helpful (for instance if another agent needs a door open but their hands are full, the system has nothing urgent to do and is near the door) and when not (the system comes to believe that an agent in Australia needs a door opening). Rather complex issues of politeness vs. resource usage make these decisions very hard and so we do not attempt to provide a solution here.

These rules *do* reflect some simple intuitions about how agents must behave in simple discourses. The speaker is responsible for guiding the audience's choice of interpretation, to the extent that it affects the speakers goals, since such interpretive actions will be inserted in support of the speaker's goals. However the audience is responsible for tidying up any side effects of interpretation, for instance the audience will be assigned responsibility for closing discourse contexts since the rule that they must be closed will fire on the audiences beliefs.

§5.4.4 Observing the Actions of Other Agents

Although taking into account the actions of other agents is important in a multi-agent domain, as important (and central to problems involving communication)

is persuading other agents to perform actions which they would not otherwise perform and which help achieve your goals.

One of the most important class of situations in which this must be done is when it is necessary to change another agent's beliefs. No sensible set of actions by the system would enable it to change the beliefs of another agent directly since to do so would involve directly interfering with another agent's internal state, which we take to be an impossibility. In such a case, the only possible course of action is to have the other agent perform an action which will change their own beliefs.

Obviously there is a problem in that with no ability to change an agent's beliefs there is no way to influence their choice of action. In human/human interaction (and indeed in human/machine interaction) this problem is avoided because the agents involved are constantly aware of changes in their surroundings so one agent can change the environment in the knowledge that another agent will notice this change, and modify their beliefs accordingly (this might be seen as a simplified version of Sperber and Wilson's concept of 'ostensive' behaviour [Sperber and Wilson 1986]).

To give this ability to the system we assume that in describing a domain it is possible to distinguish between two kinds of action —

Visible Those actions whose performance is obvious to all agents.

Invisible Those actions which, when performed, are only visible to the agent performing them.

Examples of visible actions are making a statement, moving a block or ringing the bell on the terminal (for an agent who is a computer). Making an inference or accepting a fact are invisible actions.

Invisible actions can be treated just as has been described until now. They will have pre- and post-conditions in the plan of the agent performing them and possibly in 'lower' plans, as shown in figure 5–53 but not in the plans they believe others to be following. Visible actions however must have effects on the plans of all agents. To do this we extend the procedure for inserting an action into the plan as follows: when inserting a visible action for agent 'a' to perform, insert the

same action (and its pre- and post-conditions) into agent 'a's model of every other agent's model of agent 'a's plan[4].

An example will probably make this clearer. In a domain with only two agents, the system and 'Mary', the system plans to utter the phrase "Hello". Making an utterance is a visible action and so the resulting plan fragment might look like figure 5–54 (omitting the circular 'believe' facts for clarity). We have included all the information which would be contained in the views so as to show how the conventions given earlier about the self-concept 'ME' would structure the mappings in a reasonably complex situation. The shaded boxes in the diagram stand for the entire view which is expanded above them.

The bottommost pair of views in figure 5–54 and the associated action are part of the system's plan and so the mappings interpret the entities of the descriptions as real entities in the domain (here we assume 'Mary' and 'system' are such entities, standing for Mary and the system itself respectively). The pre- and post-conditions of the action have been copied to Mary's model of the system's plan so that Mary is seen to know what action has been performed and what the system expects the results to be; this gives rise to the topmost pair of views and the copied action. Finally Mary is shown to believe what she believes the relevant pre- and post-conditions will be, here for simplicity we assume that Mary and the system do not disagree about the outcome of the action so the middle pair of views, representing Mary's beliefs are simply copies of the top pair except that the entities are re-interpreted.

§5.4.5 Planning Actions for Other Agents

With the ability to have some effect on another agent's beliefs as described in section §5.4.4 the system has all the tools necessary to actually make other agents perform actions on its behalf. All that is necessary is that we define the precise conditions under which an agent will perform an action so that the system can

---

[4]Since visibility is a property of classes of actions (for instance all instances of 'put-on-table' are visible) it is not necessary to represent their visibility within the plan. All that need be done is to make sure the pre- and post-conditions of visible actions are of the format described here; later planning can treat them in the same way as any other action.

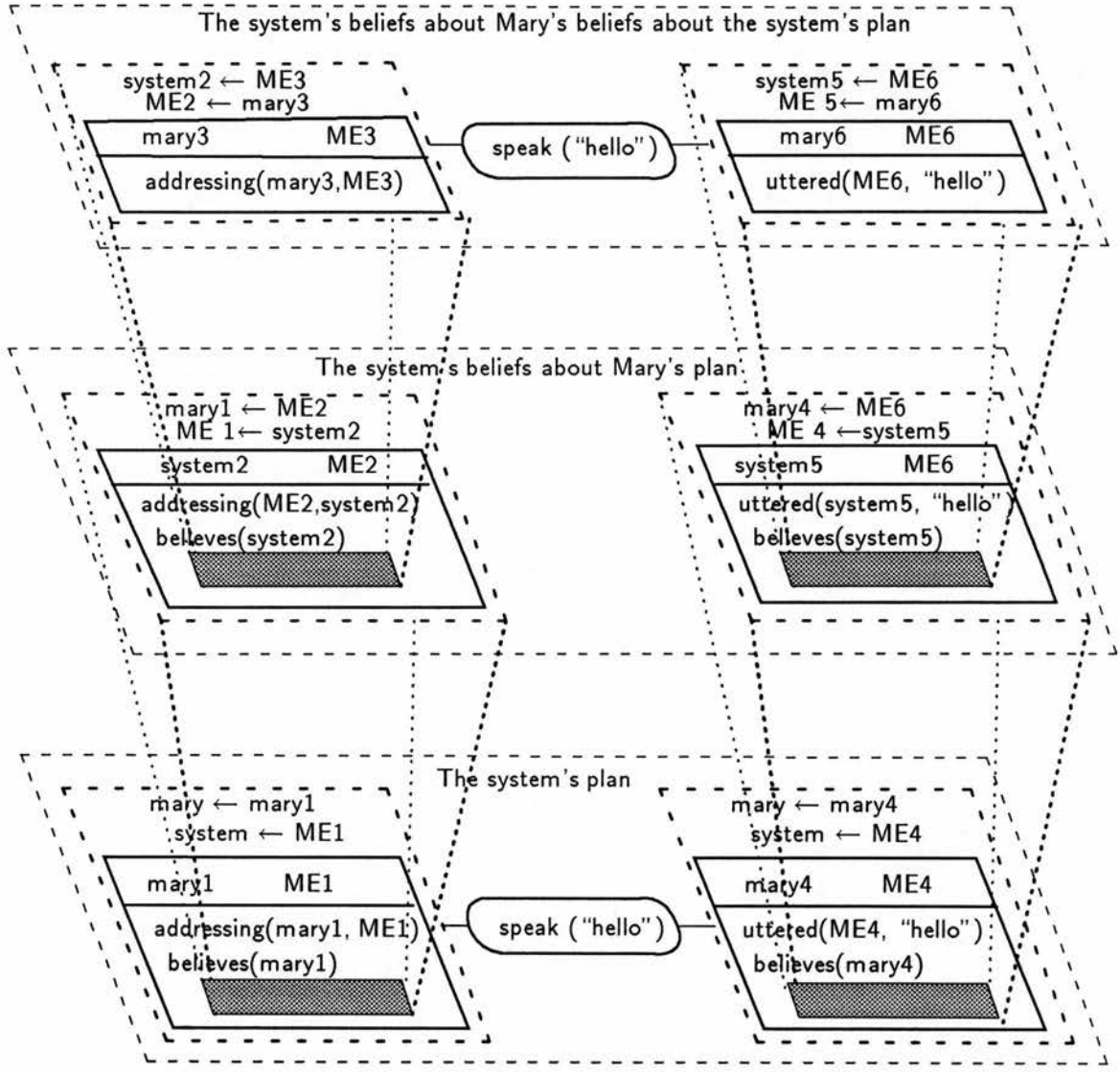


Figure 5–54: Effects of a Public Action

bring them about. One possible way of doing this would be to add a precondition to each action that the agent "want to do" that action (for instance [Rosenschein 1982] introduces a predicate `Will-Perform(agent, action)` for this purpose). The following is such a set of conditions —

Useful The agent must have a goal which the action can bring about.

Applicable The action must be usable in the correct position of the plan, there must be no preconditions which are unsatisfiable and no conflicts which cannot be removed by restricting the ordering of the actions.

Best The action which is to be performed must be the best available action to achieve the goal. The desired action must be the one which the agent will insert into its plan in preference to any others.

When these three conditions are satisfied we say that the action is "motivated" for that agent.

These conditions can be seen as parallels to the 'relevance' criteria of Sperber & Wilson. The usefulness of an action being parallel to an utterance's implicit guarantee of its own relevance and the other two parallelling their definition of relevance as maximal effect with minimal effort. Thus a speaker attempting to motivate some (interpretive) action by an audience will be attempting to ensure the relevance of that interpretation to that audience.

Any action which the system plans to perform itself will be motivated for the system since the three conditions will be guaranteed by the planning algorithm, but when the system plans an action for another agent we must be sure that it fulfills the conditions.

There are two ways in which a system of the type described here can receive a goal —

- The firing of a rule.
- As a precondition of a planned action.

The system can arrange the first sort of goal by planning to create a situation in which the beliefs of the agent will match the pattern of the rule. This is simply a case of inserting a state into the plan with the necessary beliefs stated and then the usual process of finding support will guarantee the situation comes about. The second variety of goal can be provided by recursively applying the conditions here to make the agent plan an action for which the desired goal is a precondition. Obviously this raises a problem with cycles in the behaviour of the system. However since the domain of the system is always finite there will be no case in which there are an infinite number of possible actions and so it is simply a case of ensuring that the system never tries to force an action as part of the process of forcing that same action.

Arranging for the desired action to be applicable is not a problem since as we insert the action into the plan it becomes part of the normal planning process to ensure that it does not conflict. Similarly the rules set out in section §5.4.3 ensure that it is considered the system's responsibility to provide support for the preconditions of the action (since the action is inserted into the plan to support a complex term which is the system's responsibility).

Finally the system must ensure that the action is the one considered best out of all those which might be available. This is only really possible because we are making the assumption that all agents are of the same type as the system (at least in performance, if not in implementation). Given this assumption the system can simply choose an action for the desired goal using its own heuristics and then, if it is not the correct one, make sure that this action cannot be used, by removing some of its preconditions for instance, pick another and so on until the correct one is selected. If the other agent is of the same type as the system this will ensure that in applying its heuristics it will arrive at the correct action.

This assumption of complete agreement between agents as to what actions should be chosen in what circumstances is obviously extreme. There is no doubt that in real situations human beings make do with far less certainty as to the plans of others. However when planning communicative actions it is hard or even impossible to take into account all the ways an utterance might be interpreted by a sufficiently atypical audience and it seems there must exist strong expectations about the interpretations which the audience is most likely to make. It would seem that an agent planning communicative actions *must* make such assumptions and rely on discourse repair strategies when they fail.

Given this ability to make other agents perform actions, we must now specify when it will be useful and how it is to be applied. Planning an action for another agent will be another possible fix for chores of type '**Support an Unsupported Fact**'. When we try to apply this fix for an unsupported complex fact 'F' we must perform the following steps —

- Select an agent 'A'.
- Find a fact 'F''' which, if believed by that agent would mean that the unsupported complex fact is supported.
- Select an action which if executed by the selected agent will cause 'F''' to hold.
- Use the above definition of motivation to motivate the selected action for 'A'.

In general there might be a number of choices at each stage and so a number of possible fixes.

The only new concept here is that of 'a fact which if believed by an agent would make a given fact true'. This is simply a way of saying that when selecting an action for the other agent, the system must take into account the rules for inserting actions by other agents into the plan described in section §5.4.2. For example, if the system wishes to support '**blue (car1)**' by having Mary perform an action but Mary believes car2 is car1 then the fact which Mary must plan for is '**blue (car2)**'. Similarly if the system wishes to support '**believe (mary), blue (car1)**' then it is enough for the system to manoeuvre Mary into performing an action which she sees as resulting in '**believe (ME), blue (car2)**'. This second example represents a simple example of communication since it is not necessary that the action performed by Mary result in car1 actually being blue (in the system's beliefs) only that Mary believe it to be. This could be done by the system performing a communicative action of some kind if Mary could interpret it correctly. This type of planning underlies the discourse planning behaviour discussed in chapter 7.

§5.4.6 Executing Actions

When an action in the system's plan which is to be performed by another agent becomes eligible for execution — that is, when its pre-condition is the current

state of the world (in fact its precondition will be the view which describes the agents beliefs in the current state of the world) — the system must do something to determine when it has been executed and what the result was.

The simplest assumption is to use the method described for "symbolic execution" in section §5.2.3.5, that is change the current state of the world to conform to the expected post condition of the action. When this is done the system can just wait until it sees (by whatever means it has of observing the outside world) that the expected conditions have come about.

This method has very obvious shortcomings. There seems no way to cope with actions by other agents which do not have the expected effect or which do not happen at all. In such a case the system would just wait forever. Worse in many ways is the fact that the system cannot cope with actions which happen in an unexpected order; it must commit itself to one order and await the execution of the first. If the two actions are totally independent then there is no problem, since executing the first will be unnoticed but its results will be in effect after executing the second when the system comes to look for them. If two interacting actions are executed in the opposite order to that which the system expects then it will, once again be stuck waiting for a condition which will never occur.

Despite these limitations, this simple solution is the one we will assume here. In planning for simple communication we can sidestep the harder problems of execution monitoring since the actions we expect from the audience will all be private. If the audience accepts a fact as true or closes a discourse context, there will be no direct visible result and so no simple way of checking that the desired action has been performed. If this work were to be extended to cases of simple dialogue, for example, this is one of the areas where a great deal of work would need to be done.

§5.5 Summary of the Planning Mechanism

Having introduced a number of extensions to the basic planning system of sections §5.1 and §5.2 we now summerise the operation of the entire extended system.

The plans manipulated are constructed from views, described in section §4.3. Each state in the plan is represented by a tree of such views, a subset of the nodes in this tree, itself forming a tree with the same root will be composed of 'belief views' which represent the beliefs of some agent which will obey the constraints

of section §5.4.1. The relationships between states of the plan are represented by giving a start and end point for each state; these points may be either fixed or movable. States in the plan are marked as the responsibility of an agent when they are inserted according to the rules given in section §5.4.3.

The system's main behaviour is to repeatedly examine the plan to determine if any of the following faults can be found with it.

1. There exists a belief view as part of a state in the plan which matches the left hand side of a rule and the corresponding instance of the right hand side of the same rule does not exist later in the plan.

2. There exists in a state in the plan which is the responsibility of some agent 'a', an unsupported complex fact in a belief view representing the beliefs of agent 'a'.

3. The plan contains an action to be performed by an agent 'a' which is not motivated for 'a' according to the rules of section §5.4.5.

4. An action in the plan is eligible for execution, or a state is eligible for incorporation into the initial state of the plan.

5. An unbound place holder exists in the plan.

If one of these faults is found the system must apply a fix. The fixes are essentially those of section §5.2.3 except that complex terms may be supported by inserting actions by other agents into the plan in the manner described in section §5.4.2 and extra types of fix must be included for providing motivation and for firing rules. The latter is simple, the required instance of the right hand side of the rule is inserted into the plan. The former is more complex.

Motivation is providing by ensuring that the clauses of the definition in section §5.4.5 are true. The only problematic clause is the one labelled 'best'. The system must ensure that the action planned for is the one which the other agent will think is the best. It can do this so long as it knows the order in which the other agent will examine the possibilities, which we have assumed it can.

Some actions are known to the system to be 'visible', that is the fact that they have been performed and the preconditions and results of their performance are known to all agents. Such actions have pre- and postcondition of the form described in

section §5.4.4, consisting of repeats of the action in other agents' models of the performing agents plan.

Chapter 6

Riple, a Simple Multi Agent Planner for Text Planning

§1 Introduction

This chapter describes Riple, a simple planning system for domains involving other agents based on the ideas developed in chapter 5.

Riple is given a description of a planning domain containing action descriptions and rules for when to adopt goals and is started by giving it a description of a state of the world. It then derives goals and plans actions for itself and other agents. When planned actions become executable it 'executes' them. 'Execution' of actions for other agents is simply implemented by printing a message saying that the system expects a certain event to occur. 'Execution' of actions by the system is also implemented by simply printing a message, though there is provision for attaching procedures to actions.

Since Riple is a concrete system certain choices had to be made in its implementation where issues were left open by the discussion in chapter 5. In this chapter those choices are described and explained, though no claim is made that those choices will be the best for every possible domain. The development of Riple has been guided by the needs of text planning and specifically by the desire to produce a system capable of encoding and using simple audience interpretation strategies such as the 'points' model presented in chapter 2.

In section §2 we describe what the Riple system demands of the representation used to represent states in the plan. Then in section §3 the notation which is

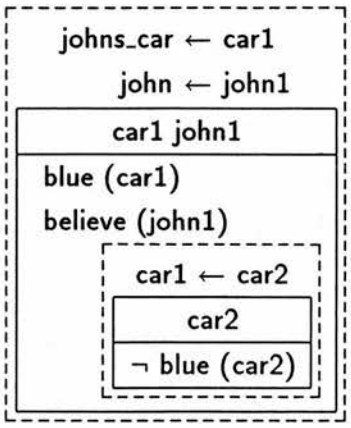


Figure 6–1: A Pictorial Description of a State of the World

used to describe planning domains to Riple and its relationship to the underlying representation is defined. Finally, section §4 describes the planning algorithm and the choices made in its implementation.

Examples of the use of this system to implement text planning systems will be presented in chapter 7.

§2 Representing the World

In sections §3 and §4 of chapter 5 we developed a representation capable of representing multiple descriptions of the world and the relationships between entities in those states. This representation forms the basis of Riple's representations of states of the world in its planning process. A small example of such a representation is shown in figure 6–1 using the pictorial notation developed in chapter 5.

This representation scheme is very general as it allows any form of description which can be created by assigning true or false to terms to be linked together by arbitrary mappings between their entity sets. However the structures which can be usefully manipulated by Riple's planning algorithm are rather more limited. Certain assumptions made in the development of the plan manipulations and the model of interaction with other agents imply that only certain forms of structure are useful to represent states in a plan.

§2.1 One to Many Relationships

Although the full representation developed in chapter 5 allows any form of relationship between the entities in a view and those of the surrounding context, it is in fact the case that for the types of model we develop in chapter 7 only relationships which give each entity in a view a unique interpretation in the surrounding context are needed. For purely implementational reasons this restriction was carried through into the actual system.

This restriction means that we cannot represent certain types of confusion of entities of the sort discussed by Fauconnier [Fauconnier 1985]. The cases we *can* represent are those where the system comes to conflate two of the entities in its model of another agent's beliefs, for instance when it is told that two people who had been previously described to it are in fact the same person. In such cases both of the old concepts are given unique interpretations in the new context, though this interpretation happens to be the same in both cases. The reverse case, where one concept is found to be a conflation and must be separated, and similar problems are much harder and the ability to represent the split would not be useful without a mechanism to deal with the complex issues which arise. Since we have no theory

of how to deal with this kind of event, the inability of Riple to represent the bare mechanics of the split is not a heavy restriction.

§2.2 Belief Views

The domains which Riple must deal with contain a certain amount of structure, and some of this structure will be reflected in the kinds of representation which will be produced. The most important structural constraint on the domains is the existence of one or more active entities, called agents. Agents have beliefs and those beliefs include beliefs about themselves and other agents. This imposes a tree like structure on the set of views which make up a world description in Riple. Additionally there are, as described in section §5.4.1 of chapter 5 certain assumptions we can make about agents' knowledge of themselves and their beliefs that impose more restrictions on the form of world descriptions. These restrictions will be examined in more detail in section §3.2 when we describe the syntax which can be used to define world descriptions for Riple.

§2.3 Constants, Agents and Place Holders

In languages designed as formal models of the world, such as logics and the input languages of planning systems, it is usual to make a distinction between constants and variables. Constants represent 'real' things, while variables represent unspecified things which other parts of the description may define further. The representation we have so far developed contains no such distinction, though the planning model of section §5.1.3 of chapter 5 introduced the concept of a place holder which serves a similar purpose to that of a logical variable.

The uniform concept of an 'entity' in our representation is convenient when we are thinking of it only as a representation. However when we wish to define a planning system it is useful to distinguish between three classes of entity which may occur in the world descriptions making up the plan and descriptions of actions and rules.

Constants are entities which represent globally known objects in the domain. In effect they represent objects whose identity is not in dispute, at least for the purposes of a particular planning task. For instance in the blocks world, the table and probably most of the blocks would be constants since there

is little interest in letting agents disagree about their identity. In a linguistic model, entities such as words and tenses might be assumed constant while objects in the domain of discourse might have their identities negotiated between agents.

Agents are the active entities in the domain. Apart from their special status to the planning algorithm these can be thought of as simply a subclass of constants.

Place holders are the remaining entities. Their identity is subject to change and interpretation by the planning process.

The main distinction between constants and agents on the one hand and place holders on the other is that the identity of a constant or agent is somehow common to all agents and so can be used as an anchor in dealing with other agents. There are a number of ways in which this idea could be implemented within the general representation, giving different models of what it means to be 'commonly known'.

In Riple any constant 'c' is associated with a predicate 'c_p' which provides a necessary and sufficient condition of their identity. Since predicates are global in our representation, constants will be uniquely identifiable in any context where the predicate is asserted of it. In addition to this we stipulate that in any context which represents the beliefs of some agent, the identity term of every constant used in the context will be asserted true of that constant.

This interpretation of what it means to be a 'constant' makes all constants into 'rigid designators' [Kripke 1972].

§3 A Language for Describing Domains

The pictorial description of world descriptions we have developed so far are not well suited to the task of describing a planning domain to a planner. Apart from their reliance on boxing and indentation to show the structure, which could be removed by providing a simple syntactic equivalent, they show the structure of the representation in a far more detailed way than is reasonable in a domain description. Additionally, it allows structures to be described which are outside

the restrictions described in section §2 which would lead to domain descriptions which could not be understood by the planning algorithm.

In this section we describe the syntax of domain descriptions actually understood by Riple. This syntax was designed to enable a person writing a domain description to use a relatively transparent notation which hides much of the detail of entities, views and the relationships between them. The interpretation of this notation results in the construction of structures describing actions, rules and so on which when interpreted by the Riple planning system will result in behaviour consistent with the intuitive reading of the notation. In this way this notation acts as a bridge between Riple's notions of multi-agent planning and belief representation and the linguistic intuitions embodied in interpretation models.

§3.1 Declarations

As was described in section §2.3, it is useful in describing a domain to think of certain entities as 'constants'. The idea described there of associating a predicate with each constant and asserting the identity of every constant in each belief representation in which it is referenced would be tedious to carry out by hand and so we wish the planning system to perform it for us. Also some entities in a domain will be agents able to perform actions of their own and so must be known to the planning process. For this reason we demand that the constants and agents used in a domain description be declared at the start of the description. This has the additional advantage of making it easy to catch certain common classes of typing error in domain descriptions. For this latter reason we also demand that functors to be used to build terms in the domain be declared as to their arity and their type (boolean, modal or descriptive).

The syntax of the constant and agent declarations is very simple. For instance —

```
constants
    Johns_car
    Past_tense
    ;

agents
    John
    ;
```

Functor declarations are a little more complex —

```
functors
    boolean
        Blue/1
        On/2
    modal
        believe/1
        current_context/0
    descriptive
        agent/1
    ;
```

Which declares two boolean functors, `Blue` which has one argument and `On` which has two, two modal functors, `believe` of one argument and `current_context` of none, and a one argument descriptive functor `agent`.

Because they are needed in any non-trivial domain, the constant `planner` and the modal function `believe` are defined automatically by the system. There is also a special constant `ME` which is pre-defined, but whose meaning is rather special, see section §3.2 below.

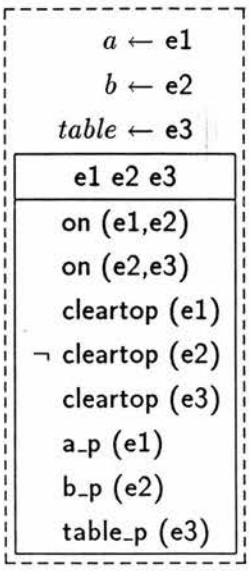
§3.2 Views, Contexts and Outlines

The heart of the domain description language is the syntax it provides for describing states of the world. This syntax must be translated by Riple into sets of views with the relationships between entities in the views defined to represent the intuitive meaning of the descriptions. Since we do not wish to force the domain writer to include details of entities and their mappings to other entities in the description, the syntax is under-specified and the system relies on context and conventions to determine the details of the concrete representation. For this reason we call the syntactic structure an 'outline'.

The following simple outline might be used to represent the initial state of a blocks world problem —

```
on (a,b)
on (b,table)
cleartop (a)
cleartop (b)
cleartop (table)
```

The interpretation of this outline is fairly obvious —



Where the italicised entities in the mapping of the view will be determined by the surrounding context in which the outline is found, as described in more detail later.

Notice that the three constants are simply represented by entities in the context of the view which are asserted to have the property of being the constant entity in question. If this outline represents the beliefs of some agent then there will be some additional information included, see section §3.2.4.

§3.2.1 The Full Syntax of Outlines

The general form of an outline is —

[*list of quantified place holders*]
fact
fact
...

The square brackets may be omitted if there are no quantified place holders.

'Facts' in an outline may be either boolean or modal. A boolean fact is a term built from a boolean functor and the correct number of constants or place holders, optionally preceded by a tilde '~' indicating negation. A modal fact is a term built from a modal functor and the correct number of place holders or constants followed by an equals sign '=' and an outline inside braces '{}' which describes the value assigned to the term. The following are legal facts —

```
  on (a, b)
~ blue (?car)
  believe (?someone) = {
                       [ ?a ]
                       on (johns_car, ?a)
                       }
```

The first two being boolean facts and the third being modal one.

§3.2.2 Outlines and Parents

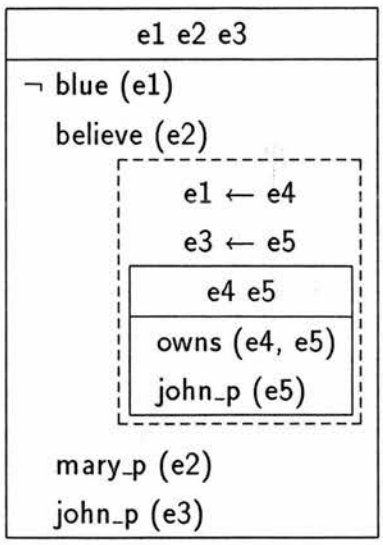
An outline represents a view of the kind defined in section §4.3 of chapter 5, that is an assignment of values to terms and a mapping giving interpretations to entities. It is fairly obvious how the facts listed in the outline can be used to construct the context in the view. What remains to be specified is the mapping between entities in that context and those in the surrounding context.

It obviously does not make sense to talk about such a mapping without knowing what the surrounding context is. For outlines which occur as the value of a modal fact, the surrounding context is obvious, for those in other places (for instance those which form the preconditions and results of an action) it is less so. For now we assume that for any outline we can identify a suitable parent outline or determine that it has none.

For an outline with a parent the mapping is constructed by mapping the entity associated each constant or place holder in the outline with the entity associated with the same constant or place holder in the parent outline. Notice that this may involve introducing such an entity into the parent if the appropriate place holder or constant was not explicitly referenced there. For example, assume the following nested outlines appeared in, say, an action description —

```
~ blue (?car)
  believe (mary) = {
                   owns (?car, john)
                   }
```

These would be represented as follows (only the context of the view representing the outer outline is shown since we do not know what its parent looks like) —



Notice that an entity for '`john`' has been included in the outer context even though '`john`' is not directly referenced in the outer outline. This representation is not quite complete, section §3.2.4 adds more information to give the required semantics to the `believe` term.

An outline with no parent represents a direct model of the world in some way and its interpretation is provided by the system's actual concepts of various entities. For this reason the mapping for such an outline is constructed by mapping the entities associated with each constant in the outline with an entity which the system maintains for that constant. Place holders in top level outlines are not mapped to anything. As will be shown in section §3.2.3 this means that all such place holders are, in a limited sense, universally quantified.

§3.2.3 Quantified Place Holders

As was discussed in section §5.3.3 of chapter 5 it is useful to be able to indicate a simple form of universal quantification in the domain description. Syntactically this is done by placing a list of quantified place holders at the start of an outline. Such a list simply indicates to the system that the entities associated with those place holders in the context constructed from the outline should not be mapped to any entity in the surrounding context. For example —

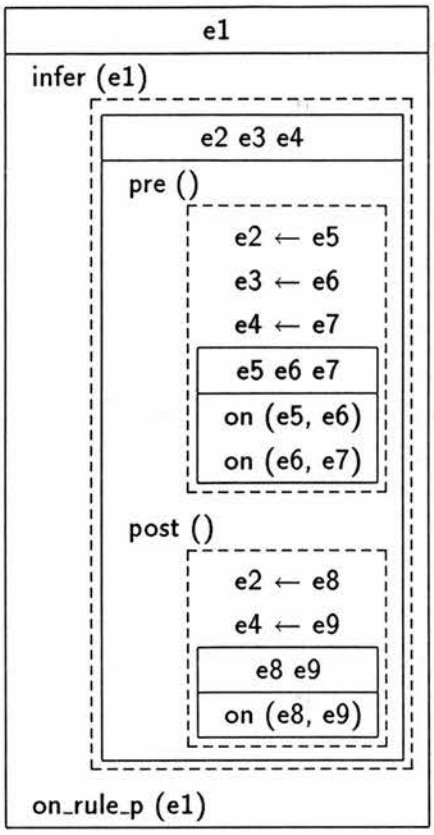


Figure 6–2: The Representation for an Outline with Quantification

```
infer (on_rule) = { [?a ?b ?c]
                  pre() = {
                          on (?a, ?b)
                          on (?b, ?c)
                          }
                  post() = {
                           on (?a, ?c)
                           }
                  }
```

which might be used to represent a simple inference rule would be represented by the structure shown in figure 6–2

And so the mappings would ensure that the matching place holders in the pre and post conditions of the inference rule are the same entity while the quantification ensures that those place holders are not identified with any real entity when the rule is used.

§3.2.4 Outlines and Agents' Beliefs

Some outlines represent the beliefs of agents in the domain. 'Top level' outlines represent the beliefs of the system itself and any outline which is the value of a modal term built from an agent '`a1`' and the functor `believe` in the beliefs of some agent'`a2`' represents `a2`'s beliefs about `a1`'s beliefs.

As was described in section §5.4.1 of chapter 5, we call the views associated with such outlines 'belief views'. All belief views must have the following two properties —

Ownership Every belief view will contain an entity which represents the agent whose beliefs it represents. For a view constructed for a top level outline this will be the agent `planner`, for a view constructed to be the value of `believe (a1)` it will be `a1`.

Reflexivity Any agent is assumed to have full knowledge of their own beliefs. That is, an agent's beliefs and their beliefs about their beliefs are indistinguishable.

Ownership is implemented by introducing into every belief view an entity representing the owner. If the view was constructed as the value of a belief term, the agent in that belief term is mapped to the owner entity by the mapping of the view, otherwise a global entity maintained by the system as its representation of itself is used. The pseudo-constant '`ME`' can be used in outlines to refer to the owning entity of the closest surrounding believe outline.

Reflexivity is implemented by ensuring that the context of a belief view assigns a view consisting of that context and the identity mapping to the term `believe` (*self*) where *self* is the owning entity of the view.

As an example of these rules, consider again the simple outline

```
~ blue (?car)
  believe (mary) = {
                   owns (?car, john)
                   }
```

The full representation for this would be as shown in figure 6–3 where the double walled boxes 'inner' and 'outer' are used to indicate where the representation would be cyclic, containing links to the full contexts with the same label.

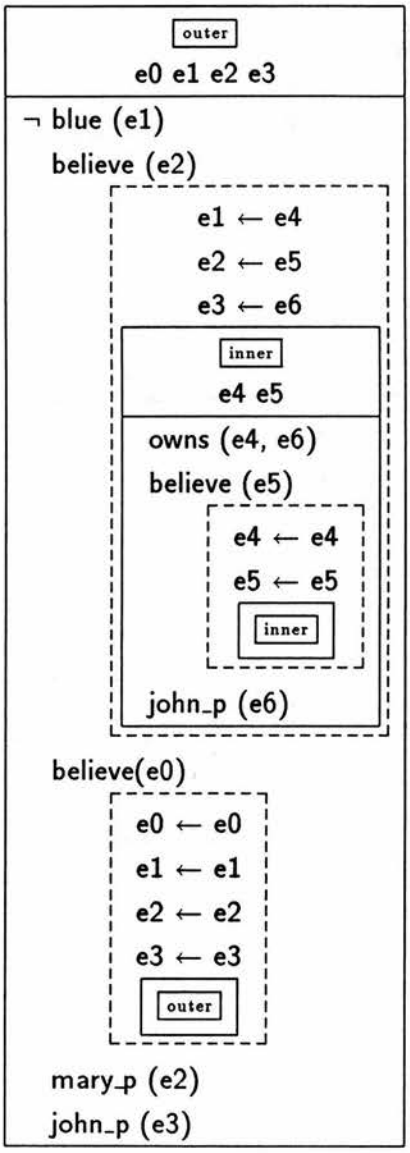


Figure 6–3: The Representation for a Simple Outline with Beliefs

§3.2.5 Outlines and Visibility

There is one final complication of the relationship between outlines and the underlying representation. The notion of 'visible' actions described in section §5.4.4 of chapter 5 requires that such actions affect the beliefs of all agents in the domain. It would obviously be possible to have the descriptions of such actions in the domain contain their full effects, but for most uses the effects will simply be to propagate the effects of the action into every agent's beliefs and their beliefs about the performing agent and so it seems best to hide these predictable details. It is, of course, still possible for someone defining a domain to describe actions whose 'visibility' does not fit this simple model by giving the effects explicitly.

The implementation of this conventional idea of visibility is simply to define the underlying representation constructed for a visible outline to be the result of constructing the representation which would be assigned for a non-visible outline with all the propagated facts explicitly given. For instance, the outline –

```
blue (car1)
believe (john) {
    wet (car1)
    }
```

would be assigned the representation shown in figure 6–4 if visible and part of a domain containing two agents, the system and `john`. For clarity this figure omits the circular modal facts enforcing the reflexivity of belief.

§3.3 Actions and Rules

Much of Riple's knowledge of its domain takes the form of action and rule definitions. Both types of definition are built from outlines of the type described in section §3.2.

An action has the following form —

`action [ visible ]` *name*
description outline
`from`
preconditions outline
`to`
results outline
`end`

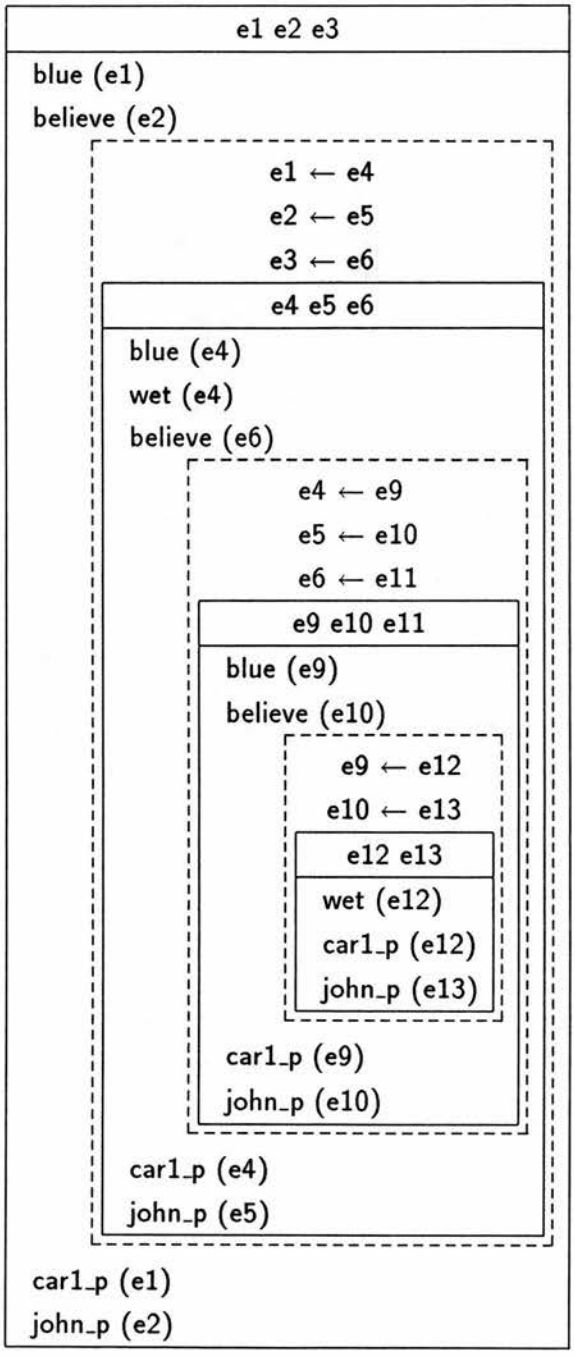


Figure 6–4: The Representation for a Visible Outline (Omitting Reflexive Beliefs)

The description outline serves only to provide a hook for the outside world into Riple. In normal operation Riple will print the description outline of every action executed with any place holders appropriately bound. Only functors declared to be of type descriptive can be used to build terms in the descriptive outline of an action and this is their only use.

The precondition and results outlines of the action describe the actual action. They are belief outlines, their ownership will be left undecided when they are first parsed and determined when instances of the action are inserted into the plan.

All three outlines have no parent outline, so their entities are interpreted relative to Riple's constant entities as described in section §3.2.2.

A rule has the following form —

```
rule name
condition outline
gives
goals outline
end
```

The condition and goal outlines are parentless belief outlines whose ownership is determined when they are inserted into a plan.

The name of the action or rule has no significance to Riple.

§3.4 Scenarios

The final type of definition needed for a domain is the scenario definition. A scenario defines a start state from which the system can be started. A domain might contain a number of scenario definitions allowing Riple to run from a number of initial states.

A scenario definition simply gives a name to a description of a possible state of the world.

```
scenario name
outline
;
```

The outline is a parentless belief outline owned by the planning system itself.

§4 How the Planning Algorithm Works

Most of the behaviour of the planning algorithm used by Riple was described in chapter 5. This section summarises this algorithm and provides information on some choices which were made to move from the theoretical model to a concrete implementation.

§4.1 Starting up

Riple is started by giving a command of the form

`plan` *scenario-name*

It begins by creating an initial plan whose start state is a copy of the view assigned to the name by a scenario definition and whose only other state is an empty view as an end state. The 'latest' pointer of the start state points at the end state and the 'earliest' pointer of the start state points at the end state. Once this is done the system enters its main loop described below.

§4.2 Constraints, Chores and Fixes

As was described in chapter 5, the main loop of the planning process consists of a continuous search for faults in the plan and attempts to fix any faults found. Riple considers a plan to be faultless iff all the following conditions hold —

1. For each state in the plan which matches the left hand side of some rule, there is a later state which matches the right hand side of that rule under the same binding of place holders to entities. See section §5.4.3 of chapter 5.

2. All complex facts in the plan are supported. That is, every complex fact is sure to be true at the point in the plan where it is encountered. See section §5.2.3.1 of chapter 5.

3. All actions in the plan are motivated; that is, every action has some reason for being there. For actions to be performed by the system this condition is trivially met. For actions to be performed by other agents there must be

some fact in the other agent's plan which would be unsupported were this action omitted and the chosen action must be the 'best' one to support that action. See section §5.4.5 of chapter 5.

4. All place holders in the plan have been bound, see section §5.2.3.3 of chapter 5.

5. Any state in the plan which could be merged into the start state has been, see section §5.2.3.4 of chapter 5.

6. All actions which can be executed have been executed. See section §5.2.3.5 of chapter 5.

Each way in which the plan fails to meet these criteria is added to an agenda of chores which must be performed.

Once Riple has examined the plan for faults it can decide what to do next. If the agenda is empty then the plan has reached a perfect state and the system can halt. Otherwise the system will choose a chore from the agenda (see section §4.3) to work on next.

For each type of chore there are a number of types of fix, as described in chapter 5. Additionally each type of fix may have a number of possible instances for any given chore (for instance more than one action might be a candidate for supporting an unsupported fact). The system will choose one of these fixes to apply to the plan, giving a new plan.

When the fix has been applied Riple will have a new plan and can begin the process again by looking for faults.

§4.3 The Agenda, Fix Choice and Backtracking

There are two choice points in the planning algorithm, choosing which chore to tackle next and choosing which fix to apply. Both have effects on the behaviour of the system.

The choice of chore from the agenda should be made carefully so as to ensure that bad plans are discovered as soon as possible and yet to avoid over constraining the planning process. Obviously this is a difficult problem and there is certainly no simple, general solution. However, since Riple is not designed to be a general

and robust planner, but rather as an illustration of some ideas and a platform for exploring issues related to text planning, a fairly simple choice was made. Riple assigns a fixed priority to each type of chore and will always attempt to perform the highest priority chore on the agenda. The priorities are as follows (from most urgent to least) —

1. Execute action.
2. Merge state into start state.
3. Eliminate Conflict.
4. Find Support.
5. Motivate Action.
6. Fire Rule.
7. Bind Place Holder.

This makes the system keen to execute actions and wary of prematurely binding place holders. Within each class of chore, Riple prefers chores which have been introduced into the plan recently to older ones which ensures that, so far as possible, problems associated with a choice of fix are found soon after the fix is applied.

Fix types are also ordered. The order is roughly that of their presentation in section §4 of chapter 5. Thus the system will attempt to insert a hold state in preference to introducing a new action to provide support. There is also a heuristic mechanism which attempts to prevent the repeated application of very similar fixes to avoid loops. This is necessary since domains containing inference actions or the type of discourse interpretation rules we present in chapter 7 often contain actions which can be instantiated so that a post-condition is the same as a pre-condition, or the same problem can arise with a sequence of actions.

Sometimes it will be the case that there is no available fix for a chore. In such a case the system must backtrack. Riple does not perform any form of intelligent backtracking. Rather, a simple chronological backtracking scheme is used. If there is no choice point to backtrack to Riple halts, indicating its failure.

As was noted in section §5.2.3.5 of chapter 5, it is not reasonable to backtrack to a choice before the last action was executed since this would involve being able

to undo actions in the real world. This limitation, combined with the choice to make the system keen to execute actions, greatly decreases the search space of the system.

The implemented program also allows a certain amount of human intervention in the chore and fix selection mechanism. As we will note in chapter 7, the choice of which term to support next has a substantial effect on the style of planned texts. The simple heuristic rule of doing recently introduced chores first makes sense from a general purpose planning point of view, though a more intelligent rule would be better. However when investigating complex plans such as those produced for paragraph length texts it is often interesting to see what happens if chores are tackled in a different order.

§4.4 Matching and Fix Selection

Central to the planning mechanism is the matching algorithm used to find rules and actions which fulfill some condition. As we said in section §5.3.2 of chapter 5 we chose to do such matching by comparing the complex facts extracted from a state using a simple graph walking algorithm rather than by using a more general cyclic graph unification algorithm for efficiency reasons. From an implementational point of view

§4.5 Efficiency Issues

The description of the planning process in section §4.2 implies that the whole plan must be examined for faults on every iteration of the planning algorithm. In fact it is generally the case that faults are introduced into the plan locally by a fix and that faults accumulate over time until fixed. This means that the agenda from one iteration of the planning algorithm can be retained into the next and that only a small amount of the plan need be examined after each fix.

An exception to the locality of fix effects is the binding of a place holder which can have global effects, so any fix which binds a place holder in the plan causes every state in the plan which mentions that place holder to be examined for new chores.

The fact that the planner will hold over an old agenda as the basis of the new one introduces the possibility that a chore may be accidentally fixed while it is on

the agenda. Fortunately this will not cause problems if every chore type is given an additional fix type which is to do nothing if the chore is no longer necessary. Obviously this fix type is preferred to any other in all cases.

§5 Summary

This chapter has summarised the representation, input language and planning algorithm of the Riple system. Although based on general concepts of rational action and belief representation, this system has been aimed specifically towards the representation and use of simple text interpretation strategies as a basis for text planning.

The language provided for creating domain descriptions for Riple is designed to have both an obvious intuitive meaning and a well defined semantics in the more general 'views' representation scheme.

Chapter 7

Text Planning Using Riple

§1 Introduction

In the preceding chapters we have motivated and described a model of planned action in domains with multiple agents. In this chapter we apply this model to the problems of text planning.

The discussion of text planning in this chapter will take the form of developing audience models of increasing complexity and showing how Riple can use these models to plan texts of increasing complexity. Each audience model takes the form of a Riple domain defining actions and rules which are available to the speaker and the audience. The speakers actions are fairly simple in all the models as we do not attempt to describe the process of planning the surface form of a text. The audience is given actions which define a strategy for interpreting utterances made to it; the speakers task is to guide the audience's use of this strategy to attain the goals which it is set.

In section §2 we describe how the structures and mechanisms provided by Riple can be used to build a text planning system. Section §3 introduces some common definitions which are assumed in the later example audience models.

Sections §4 to §7 describe very simple audience models which can be used to plan simple texts. These models are intended to provide motivation for the rather more complex audience model introduced informally in section §6 of chapter 2 which is defined as a Riple domain in section §8 of this chapter. This audience model takes the form of a simple description of text structure and a set of rules for interpreting

utterances to build such a structure. In section §9 this audience model is used to plan texts where structure and context are important.

It should be emphasised that the production of a linguistically interesting theory of discourse structure has not been part of this research. The purpose of the discourse structure and interpretation models presented in this chapter is to provide a framework for discussing the interaction between an audience model and a discourse planner. Because of this even the most complex of our models, the 'points' model, covers only a relatively small range of linguistic phenomena.

§2 What is "Text Planning" in Riple

In section §1 of chapter 1 we defined text planning to be the task of selecting one or more linguistic actions to achieve some non-linguistic goal. The inputs to a text planning system were defined to be —

- A description of the state of the world in which the text must be produced.
- A description of the beliefs of the intended audience.
- A model of the linguistic and reasoning abilities of the audience.
- A model of the behaviour of the audience.
- A goal which the text is to achieve.

Riple provides us with a representation in which these pieces of information can be given as follows.

- The state of the world and the audience's beliefs can be described in a scenario (see section §3.4 of chapter 6).
- The abilities of the audience can be defined by actions and rules (see section §3.3 of chapter 6).
- As was said in section §2 of chapter 5, it is a basic assumption of the Riple model that the other agents can be modelled as similar to the planner itself. So, our model of the audience's behaviour will simply be the Riple planning model itself. This, naturally, also limits the types of audience behaviour we can model.

- The goal can be introduced by inserting it as an unsupported state of affairs in the system's plan.

In chapter 4 we examined work in surface language generation to determine what type of description of the linguistic actions which make up a text would be expected by such a tactical generation component. There it was noted that a "deep" case representation of the semantic content of a clause together with notes describing its focus and/or informational structure was common to a number of systems. In the audience models described in this chapter the lowest level actions will be described in this way.

By "deep" (or "semantic") case we mean the description of the meaning of an utterance in terms of the roles which it assigns to the entities it mentions. For the purpose of the models presented in this chapter we assume the following cases —

agent The entity responsible for an action or having some property. In English this case is often indicated by the entity being the (grammatical) subject of a clause or, in passive clauses, by the preposition "by".

patient An entity on which an operation is performed. Often indicated in English by an entity being the (grammatical) object of a clause.

These cases have been chosen simply for the purpose of describing the simple models of interpretation and generation to be presented in this chapter. They should not be seen as linguistic claims about English.

If we represent utterances in this manner, Riple provides structures which can be used to describe the action of making an utterance in an abstract manner. An 'utterance' action will be described by giving its pre-conditions, for instance that the speaker is addressing the intended audience, its effects and relating these to a description of the act itself in the form of a set of terms. When the system executes this action, that is when the action is at the start of the plan and the system performs the actions described in sections §5.2.3.5 and §5.4.6 of chapter 5, we will assume that the utterance has been made. In a more complete system which included a tactical generating system we can assume that it is at this point that the tactical generator would be given the terms which describe the utterance to produce the surface form and utter it.

§3 Some Common Definitions

Some definitions are basic to all the example audience models described in this chapter. The definitions in this section can be assumed to be included in all later models unless it is otherwise noted. The notation used in these examples, and all later ones is that described in chapter 6.

§3.1 The World

These definitions just define the world in which the planner is operating —

```
agents
    audience
    ;

constants
    Mary
    John
    ;

functors
    modal
        believe/1
    boolean
        owns/2
    ;
```

The following definitions describe language as it is seen by the system (and as it assumes its audience sees it) —

```
functors
    boolean
        addressing/2          ;;; Agent1 is addressing Agent2
        utterance/1           ;;; Is an utterance
        interpreted/2         ;;; Agent has interpreted utterance
        audience/1            ;;; These are case frame slots
        agent/1               ;;; used to describe a clause
        predicate/1
        patient/1

    modal
        case_frame/1          ;;; Case frame of utterance is
        meaning/1             ;;; The Meaning of an utterance
        discourse_context/1   ;;; Describes the current discourse
                              ;;; context
```

```
descriptive
    with_audience/1
    with_agent/1          ;;; these are the case slots for an
    with_patient/1         ;;; utterance action
    with_predicate/1
    ;
```

Next we define the action of making an utterance. As was discussed in section §2, we describe the action in terms of the case roles giving the content of the utterance. The prefix '`with_`' on the case roles is just used to distinguish the 'descriptive' functors used to build terms which describe the utterance *action* from the 'boolean' functors used to build terms which describe the utterance itself.

```
action visible do_utterance
    with_audience (?audience)
    with_agent (?a)
    with_predicate (?p)
    with_patient (?o)
from
    addressing (ME, ?audience)
to
    utterance (?u)
  ~ interpreted (audience, ?u)
    case_frame (?u) = {
        audience (?audience)
        agent (?a)
        predicate (?p)
        patient (?o)
        }
end
```

Notice that this action is 'visible' in the sense of section §5.4.4 of chapter 5, that is when it is performed all other agents in the domain will be assumed to know that it has occurred and what the results are. This means that the structure of any instance of this action will be as shown in figure 7–1. The effects and preconditions of the action are known to the agent performing the action, the performer's models of other agents and the performer's model of other agents models of himself.

§3.2 Basic Audience Behaviour

The basic assumption which we must make about the audience's behaviour is that they are motivated to interpret what is said to them; that is that they react to linguistic actions which have them as the target audience by attempting to

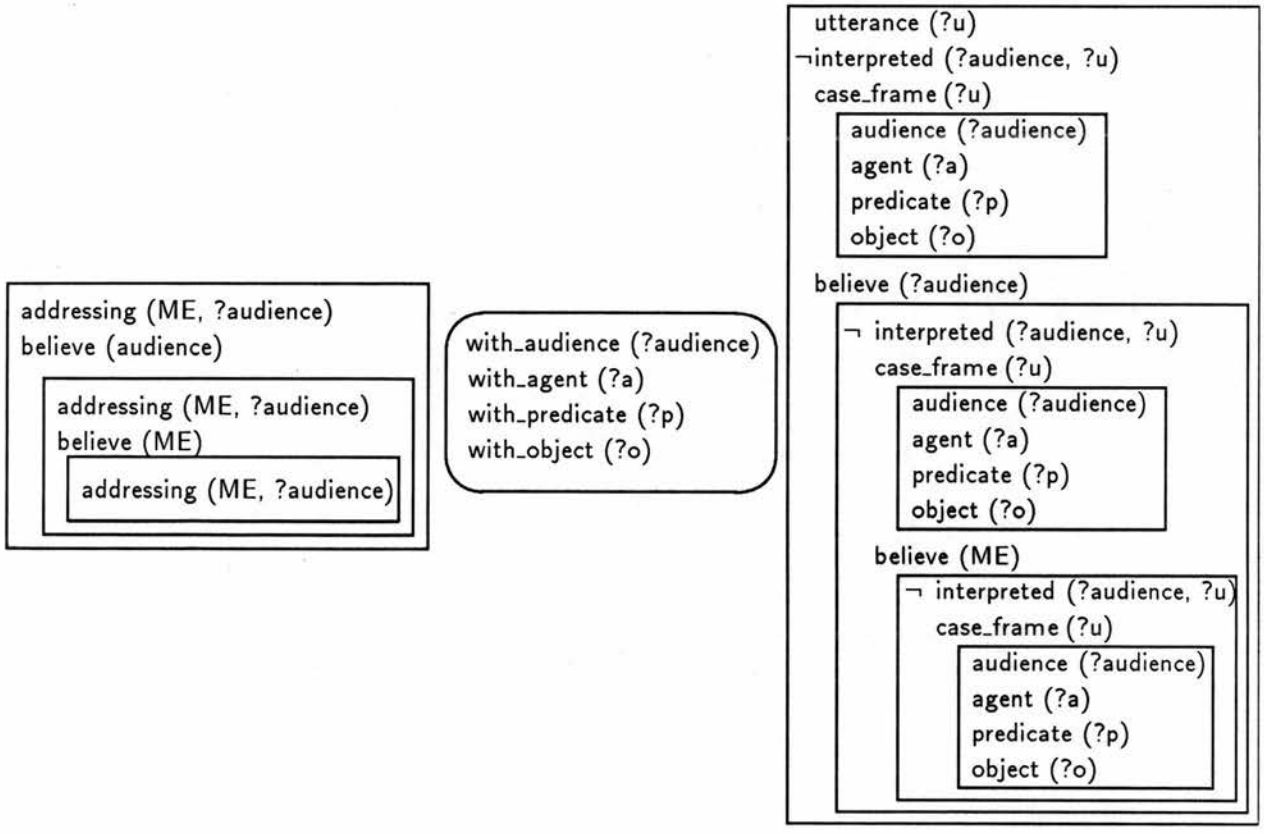


Figure 7–1: The Structure of an Utterance Action

extract some meaning. To represent this we give agents in the planner's world the following rule —

```
rule must_interpret
    utterance (?u)              ;;; If there is is an utterance
   ~interpreted (ME, ?u)        ;;; And we haven't interpreted it
    case_frame (?u) = {
        audience (ME)       ;;; And the audience is yourself
        }
gives
    interpreted (ME, ?u)         ;;; Get it interpreted
end
```

The effect of this rule is that whenever some agent's beliefs include the facts that —

- An utterance has been made.
- The utterance was addressed to the agent in question.

- This agent has not interpreted it.

then a new state will be inserted into the plan asserting that the agent believes that the utterance has been interpreted. The normal operation of Riple will then, by trying to provide 'support' for the new state, introduce new actions by that agent which will bring about the truth of the fact `interpreted (ME, ?u)`. These states, therefore, represent the goals of the agents in the domain. However in Riple no distinction is made between states representing goals and those representing effects — they are all states whose existence in the plan must be supported. Actions which can be performed to make `interpreted (ME, ?u)` true will be described in later sections.

In fact, since the models which are being described here are designed to describe utterance *production* behaviour, the rule `must_interpret` will always be used in reverse. Riple will, in trying to support some goal of its own, introduce an action by another agent into its plan which has the side-effect of making `interpreted (ME, ?u)` true. When it tries to motivate this action it will use this rule to decide on a condition which when brought about will cause the other agent to perform the desired action.

§4 A Very Simple Model

In this section we shall define a very simple generation model. This model will be extended in later sections to cope with more interesting examples.

The simplest model we might have of the behaviour of the audience is that they have a fixed repertoire of interpretation actions which allow them to move from a state where an utterance has been made to one where they believe the 'meaning' of that utterance. An example of such an action might be —

```
action interpret_owns_utterance
from
    utterance (?u)
    case_frame (?u) = {
        predicate (state_owns)
        agent (?a)
        patient (?o)
        }
to
    interpreted (?u)
    believe (ME) = {
        owns (?a, ?o)
        }
end
```

The rules defining the belief structures constructed for actions described in chapter 6 ensure that "`believe (ME)`" is an identity operation, that is that this action could equally be written with just "`owns (?a, ?o)`" in place of the `believe` fact. However the `believe` is inserted here for clarity and for consistency with later models.

With a domain consisting of just these actions and rules Riple is able to perform very simple text planning actions. As an example we shall consider a case where Riple's plan contains an unsupported complex term[1] "`believe (audience), owns (Mary, car1)`". This can be arranged as described above by including in the domain a rule which matches the initial state of the world and which has this term in its right hand side. It might also occur in the precondition of an action inserted into the plan for some other reason. In either case the behaviour of the system will be as follows.

---

[1]For a description of complex terms, see section §5.3.2 of chapter 5

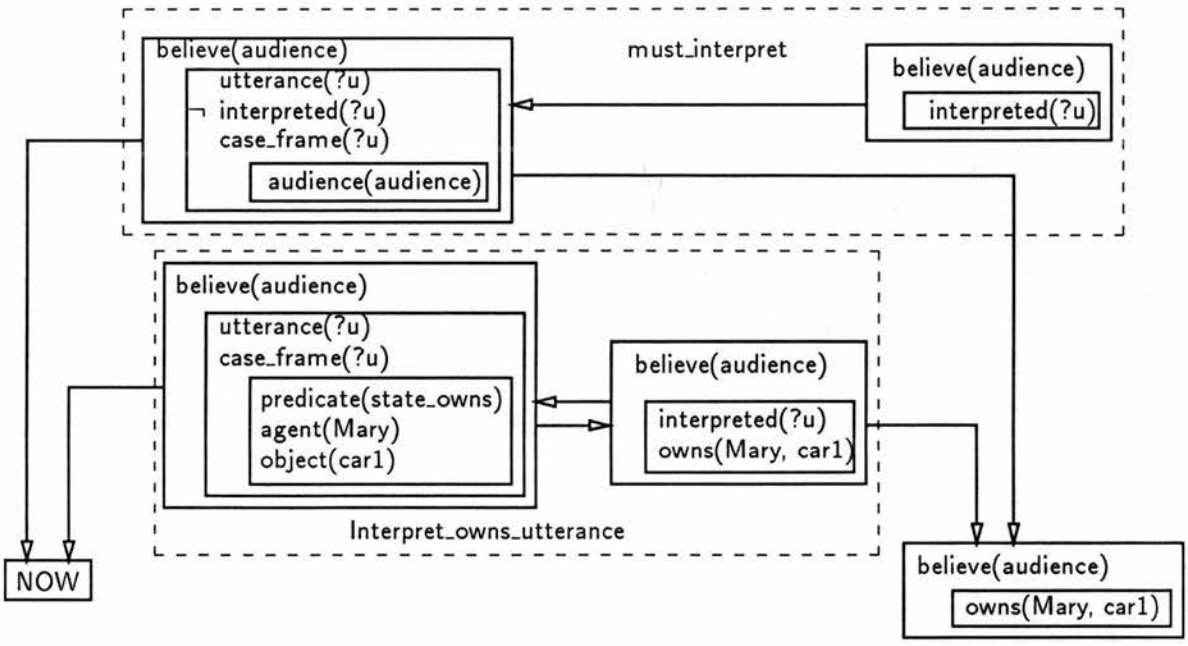


Figure 7–2: A Partial Plan For a Single Utterance

First it would plan for `audience` to perform an instance of `interpret_owns_utterance` with `?a` bound to `Mary` and `?o` bound to `car1`. To do this it will insert the pre- and postconditions of the action into the plan as beliefs of the audience and will mark the action as to be performed by the audience.

By inserting this action Riple has violated two of the conditions described in section §4.2 of chapter 6; the complex terms in the pre-condition of the action are not known to be supported and `audience` is not known to be motivated to perform the new action. Given this very limited domain, the only possible motivation for this would be an instance of the RHS of the rule `must_interpret` moved into the audience's beliefs and so Riple must support the trigger conditions of this rule, also in the audience's beliefs. At this point the plan would look like that shown in figure 7–2[2].

[2]In all the plans shown in this chapter we will omit the entity mappings which form part of the views which represent plan states. In later plans where the structure is more complex we will often omit some of the contents of the states where this is described in the accompanying text indicating this by the use of ellipsis '...'

This plan contains the following unsupported complex terms, from the instance of `interpret_owns_utterance` —

believe (audience)	utterance (?u)
believe (audience), case_frame (?u)	predicate (state_owns)
believe (audience), case_frame (?u)	agent (Mary)
believe (audience), case_frame (?u)	patient (car1)

And from the instance of `must_interpret` —

believe (audience)		utterance (?u)
believe (audience)	¬	interpreted (audience, ?u)
believe (audience), case_frame (?u)		audience (audience)

Support for all these complex facts can be obtained by inserting an instance of the `do_utterance` action.

When this action is inserted, the place holder `?u` will remain unbound; since it is in the post-condition of the action there will be no attempt to bind this place holder while finding support as the action itself provides support for facts in its post-condition. Assuming nothing else forces a binding (and there are no rules or actions in this model which could do so) it will be bound to an newly created entity when the action is executed.

The plan will now be as shown in figure 7–3 (only the initial segment is shown for space reasons).

The pre-condition of the new utterance action is simply that the system be addressing `audience`, if we assume that this is true in the current state of the world (or can be made so trivially) the plan now fulfills the first three conditions described in section §4.2 of chapter 6, all rule preconditions have matching right hand sides, every complex fact in the plan is supported and all actions are motivated. The forth condition is violated by the `do_utterance` action which is performable (it is at the start of the plan, its precondition is supported etc.) thus the system will perform this action, that is it will make the utterance. The remaining actions will also be 'performed', though since they are all planned for the audience their 'performance' will consist simply of assuming that they have been performed (none of them have an effect on the system's own beliefs, they all effect only the audience's beliefs and so the system can not wait until it knows the action has been performed, it must just assume that they have been).

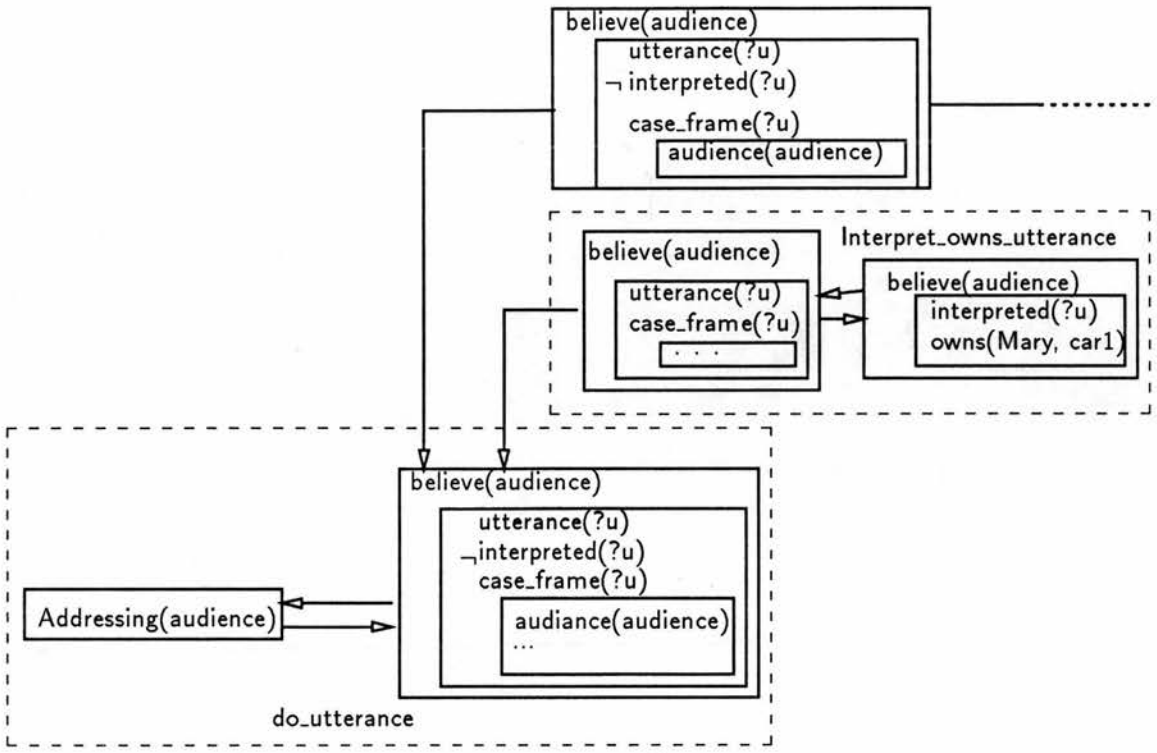


Figure 7–3: The Initial Section of The Complete Plan For a Single Utterance

§5 Discourse Assumptions and Reminders

The trivial audience model described in section §4 assumes that interpreting an utterance simply means finding a fact which could be the "meaning" of the utterance and assuming that fact. This is obviously too simplistic. In this section we consider two phenomena which can not be accounted for using this model; discourse assumptions and reminders.

Often a text will contain statements which the audience does not wish to immediately assume to be the case, for instance if making the assumption would make their beliefs inconsistent. However in such cases the audience will still wish to interpret the statements and, moreover, to keep the interpretations in mind while interpreting later parts of the text. Such facts, assumed for the sake of interpretation, we call 'discourse assumptions'. From the point of view of the speaker, discourse assumptions are placed into a text without the expectation that the au-

dience will adopt them as beliefs, for instance as part of an argument by reductio ad absurdum.

Reminders are another type of statement for which the audience model of section §4 cannot account. Reminders, by definition, are statements whose meaning is already believed by the audience and so in this simple model will be statements without effect and so, pointless. In fact reminders are common in texts of all kinds as they serve to guide the audience interpretation of the text by putting other statements in context and indicating "focus". In the simplest case a text consisting only of already believed facts can affect the audience's behaviour by "bringing something to mind".

Both these kinds of utterance can be thought of as explicitly manipulating the context of a discourse rather then the beliefs of the audience. The simple interpretation model described in section §4 can be modified to interpret, and so motivate the production of, such utterances by making the audience manipulate an explicit discourse context, rather then their beliefs.

Once again using "owns" as an example, we include in the domain the following action —

```
action interpret_owns_in_context
from
    utterance (?u)
    case_frame (?u) = {
        predicate (state_owns)
        agent (?a)
        patient (?o)
        }
to
    interpreted (?u)
    discourse_context = {
        owns (?a, ?o)
        }
end
```

This action is exactly the same as the similarly named action in the previous audience model *except* that the fact derived as the meaning of the utterance — '`owns (?a, ?o)`' — is asserted in the discourse context rather than in the audience's beliefs.

If the audience has this action available to it, the system will be able to plan to make statements which act, in a simple manner, as reminders and discourse

assumptions, that is it will be able to plan an utterance with the aim of having the audience assert a fact into the current discourse without adopting it as a belief. The construction and execution of such plans would be totally parallel to the behaviour of the previous model. However if this new definition of `interpret_owns_utterance` replaces the rule of the same name given in section §4 the system will no longer be able to plan simple information transfer actions — there will no longer be any actions available to the audience which will cause the meaning of an utterance to be inserted into their beliefs as a result of interpreting it. To overcome this we need to introduce the concept of the interpretation of a discourse.

Just as an audience must interpret simple utterances to determine why they were made, they must also be able to find interpretations for larger pieces of text. For instance, presented with the following text —

18a) Mary owns the blue car.

18b) So, Mary must be able to drive.

the audience might interpret the first sentence as a statement, the second as an inference and the combination of both as an attempt to persuade them to accept that Mary can drive.

To extend our model of interpretation to this kind of interpretation we must add a new concept to the world. A discourse must now be an object which can be explicitly manipulated and all utterances must be interpreted as part of some discourse. The following action can be used by the audience to start a discourse.

```
action start_new_discourse
from
to
    current_discourse (?d)
    discourse_context (?d) = {
        }
end
```

Notice that it is the *audience* who performs this action. The speaker will "start a new discourse" by having the audience perform this action.

This is an action with no precondition since, for the moment, we assume that an agent can assume that a new discourse has just begun at any time. We now need to ensure that the audience will attempt to interpret every discourse with the following rule, analogous to `must_interpret` —

```
rule must_close_discourse
    current_discourse (?d)
gives
   ~current_discourse (?d)
end
```

As a rule, this enforces a constraint on audience behaviour. If at any time there is a state in the plan where an agent believes `current_discourse (?d)`, for some `?d`, then there will be a later state where this is not the case. That is, every discourse which becomes current will at some time be ended. Much of the rest of this section will be concerned with the conditions under which a discourse can be considered finished.

For the purposes of this section we will assume that all the discourses we are interested in are simple lists of facts and that our audience is totally cooperative (that is that they accept anything which is said to them). In such a situation, the audience will close a discourse by accepting that the discourse context forms a true description of part of the world. The following action is the one such an audience would require.

```
action close_discourse_and_accept
from
    current_discourse (?d)
    discourse_context (?d) = ?context
to
   ~current_discourse (?d)
    believe (ME) = {
        ?context
        }
end
```

With this action, `start_new_discourse` and a version of `interpret_owns` modified to use the discourse context associated with the current discourse, Riple can support the complex fact `believe (audience), owns (Mary, car1)` as follows —

1. The system will note that the only way to have the audience believe something is with an instance of `close_discourse_and_accept` and so insert an instance of this action with `?context` bound to a view assigning the term `owns (Mary, car1)` the value TRUE. This action will be unmotivated for `audience` and will have two unsupported preconditions

   `believe (audience), current_discourse (?d)`

   `believe (audience), discourse_context (?d), owns (Mary, car1).`

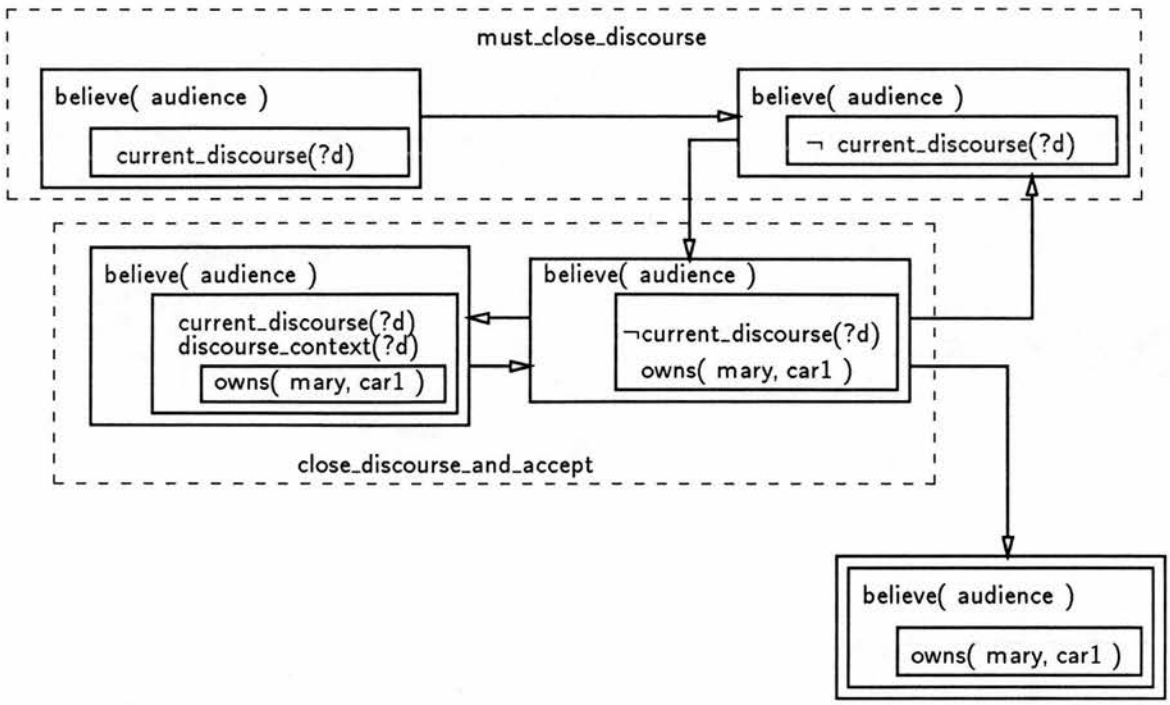


Figure 7–4: Motivating the Closure of a Discourse

2. Motivation for the audience's performance of this action can be provided by an instance of the rule `must_close_discourse`. The condition of this rule must therefore be inserted into the plan. This will cause the plan to contain another unsupported instance of `believe (audience), current_discourse (?d)`. The plan at this point is shown in figure 7–4; the double bordered box at the bottom left represents the context containing the complex term being supported.

3. Support for

   ```
   believe (audience), discourse_context (?d), owns (Mary, car1)
   ```

   can be obtained by inserting an instance of `interpret_owns` performed by the audience into the plan. The pre-condition of this action will contain the unsupported complex terms

   ```
   believe (audience), utterance (?u)
   believe (audience), current_discourse (?d)
   ```

   along with three more defining the case frame of `?u`.

4. Again, motivation must be provided for the audience's performance of this action. This can be done by adding an instance of the rule `must_interpret` to the plan in the audience's beliefs. This will require

   ```
   believe (audience), utterance (?u)
   ```

   to be supported.

5. An instance of `do_utterance` can provide support for

   ```
   believe (audience), utterance (?u)
   ```

   and also for the

   ```
   believe (audience), case_frame (?u)...
   ```

   facts. Since this is being planned as an action of the system itself, there is no need to provide motivation.

6.

   ```
   believe (audience), current_discourse (?d)
   ```

   can be supported by inserting an instance of `start_discourse` performed by the audience. The audience's motivation for this action will be the as yet unsupported precondition

   ```
   believe (audience), current_discourse (?d)
   ```

   in the audience's `interpret_owns` action.

Along with some housekeeping to get the place holders bound, this will produce a plan of the form shown in abstract in figure 7–5. Here all of the states of the plan have been labelled with 'S' or 'A' to indicate whether the action or rule of which they are a part was inserted as part of the systems plan or the system's model of the audience's plan.

In summary, the introduction of an explicit discourse context allows the system to plan discourses which contain utterances whose meaning is already believed by the audience, and to plan utterances without assuming that the audience will immediately accept the stated fact as true. That is it can plan utterances which behave *in the short term* like reminders or statements of assumptions.

However, in a wider context, the only purpose which guiding the audience in the construction of a discourse context can have for the speaker, is the eventual amalgamation of that discourse context into the audience's beliefs. That is, all

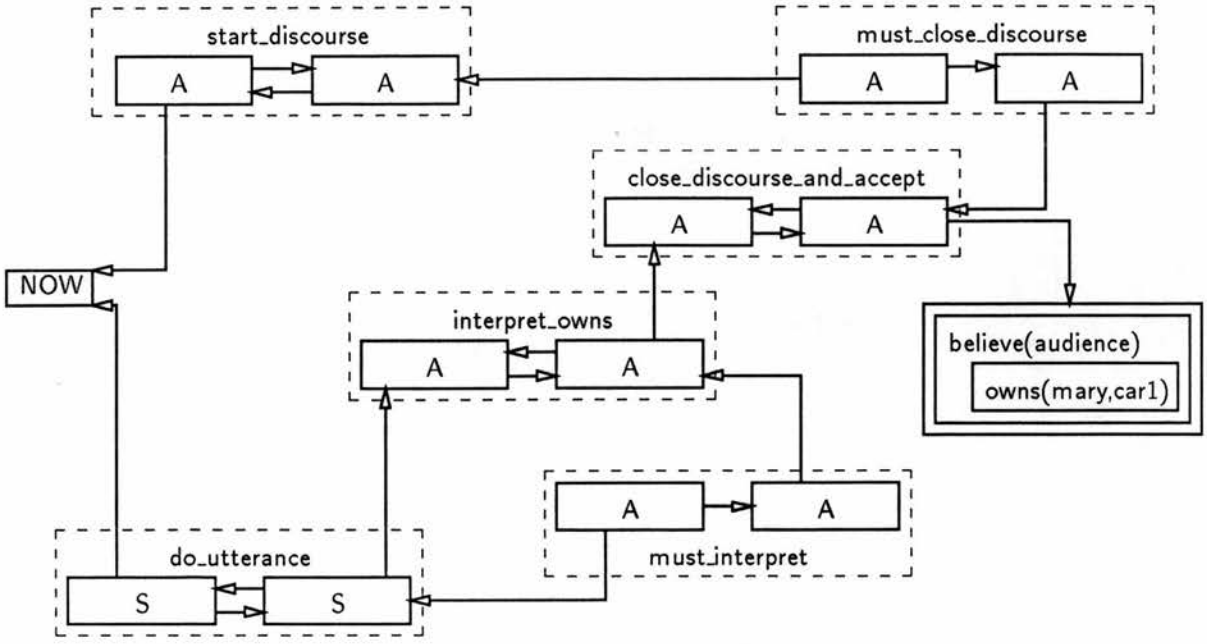


Figure 7–5: Planning to build and use a Discourse Context

utterances will eventually be treated as simple statements even if initially they have the form of reminders. To avoid this our system must be given a much richer model of the purpose of discourses. Such a model will be presented in section §6 of chapter 2. However, first we shall move on to consider another problem with this simple model, how to make the audience model more active.

§6 Guided Inference

Up to now we have assumed that we can treat the audience as totally passive and willing to accept whatever is said to them as a belief. In this section we will develop a model of the interpretation of very simple supporting arguments. By "supporting argument" we mean a text intended to make the audience accept a fact by showing that it follows from beliefs they already hold.

To make our audience model more restrictive in what it will accept as a new belief we shall constrain it to accept a fact into the discourse context only if it is plausible. Since plausibility is, in general, a very difficult concept to formalise and

one which lies beyond the scope of this work we assume the following very much oversimplified and over-restrictive definition —

A fact is said to be "plausible" in a discourse context iff

1. The fact is believed to be the case in the current state of the world.

or

2. The fact follows from facts already in the discourse context.

Notice that this definition will prevent the audience making "discourse assumptions" of the kind described in the last section. We will reintroduce these in section §7.

We represent this definition by defining as part of the domain the following two actions —

```
action believed_fact_plausible
from
    believe (ME) = {
        ?belief
        }
    current_discourse (?d)
to
    plausible_in (?d) = {
        ?belief
        }
end
```

This action is based on the first part of the above definition — if something is believed then it is plausible in any discourse context.

```
action inferred_fact_plausible
from
    current_discourse (?d)
    rule (?inference) = {
        antecedent = ?
        consequent = ?result
        }
    discourse_context (?d) = {
        ?cond
        }
to
    plausible_in (?d)= {
        ?result
        }
end
```

This action allows the result of an inference to be plausible in the discourse context if its condition is already present in the context. Once again we have extended the ontology of the system by including inference rules as entities in the world. The condition and result of an inference are associated with it by a pair of modalities. The notion of 'inference' which we are using in this model, and subsequently in this chapter, is one of plausible rather than necessary entailment. This kind of inference is more suited to the kind of reasoning necessary for interpreting discourse. Since we assume that inference is always performed in an effort to achieve some goal, rather than as and when it becomes possible, we do not suffer from the problems of combinatorial explosion which would occur were we to try and make all plausible inferences as soon as possible.

The audience's behaviour when interpreting a statement is now rather different since it is now necessary to check that the interpretation is plausible in the discourse context rather than just inserting it. Below is a new version of `interpret_owns` which behaves in this way.

```
believe (audience) = {
     owns (Mary, car1)
     car (car1)
     rule (rule_can_drive) = {
          [ ?car ?person]
          antecedent = {
               car (?car)
               owns (?person, ?car)
               }
          consequent = {
               can_drive (?person)
               }
        }
     }
```

Figure 7–6: The initial state of the world for the supportive argument example

```
action interpret_owns
from
    utterance (?u)
    case_frame (?u) = {
        predicate (state_owns)
        agent (?a)
        patient (?o)
        }
    current_discourse (?d)
    plausible_in (?d) = {
        owns (?a, ?o)
        }
to
    discourse_context (?d) = {
        owns (?a, ?o)
        }
    interpreted (ME, ?u)
end
```

In the discussion below we assume that the system has been given a domain containing actions of this form for the predicates "car" and "can drive" in addition to "owns".

A simple example of the behaviour of this model is as follows. If the initial condition is that shown in figure 7–6 and the system's plan contains the unsupported complex term

```
believe (audience), can_drive (mary)
```

then the system would plan to make the following utterances.

19a) Mary owns the blue car.

19b) The blue car is a car.

19c) Mary can drive.

(This text is, of course, *very* unnatural. See the discussion of referring expressions below and that of cue words in section §6 of chapter 2.)

To produce these utterances the system will —

1. Support the unsupported fact

   `believe (audience), can_drive (mary)`

   with an instance of `close_discourse_and_accept` to be performed by the audience and motivate this action by ensuring that an instance of the rule `must_close_discourse` is triggered in the audience's plan. This will leave the complex fact

   `believe (audience), discourse_context (?d), can_drive (mary)`

   unsupported.

2. Support

   `believe (audience), discourse_context (?d), can_drive (mary)`

   by inserting an instance of `interpret_can_drive` with the place holder `?a` bound to `Mary`. This action will have several unsupported pre-conditions of the form

   `believe (audience), case_frame (?u)...`

   and also

   `believe (audience), plausible_in (?d), can_drive (mary)`

   Throughout this explanation we shall for clarity ignore the complex terms of the form `...utterance (?u)` and `...current_discourse (?d)` which are left unsupported. They can all be supported by binding the place holders.

3. Motivate this action with an instance of the rule `must_interpret`.

4. Support the

`..., case_frame (?u), ...`

facts with an instance of `do_utterance`.

5. Support

   `...plausible_in (?d), can_drive (mary)`

   using an instance of `inferred_fact_plausible` with `?inference` bound to `rule_can_drive` and `?person` to `Mary`. The precondition of this rule will, when instantiated, contain the unsupported facts

   `believe (audience), discourse_context (?d), owns (Mary, ?car)`
   `believe (audience), discourse_context (?d), car (?car)`

6. Support

   `believe (audience), discourse_context (?d), owns (Mary, ?car)`

   by inserting an instance of `interpret_owns` with `?a` bound to `Mary` and `?o` bound to `?car`[3].

   This will leave

   `believe (audience), plausible_in (?d), owns (Mary, ?car)`

   and some

   `...case_frame (?u)...`

   facts unsupported.

7. Motivate this action with an instance of the rule `must_interpret`.

8. Support the

   `..., case_frame (?u),...`

   facts with an instance of `do_utterance`.

   At this point the plan is as shown in figure 7–7.

9. Support

---

[3]That is, `?o` and `?car` will be co-referential

`believe (audience), plausible_in (?d), owns (Mary, ?car)`

with an instance of `believed_fact_plausible`. This will leave

`believe (audience), owns (Mary, ?car)`

unsupported.

10. Support this by binding `?car` to `car1`. This is a global change to the plan — for example, in the next step it is `...car (car1)` rather then `...car (?car)` which is being supported; also this will cause `?o` to be bound, filing in the missing information in the `do_utterance` action.

11. Similarly, support

    `believe (audience), discourse_context (?d), car (car1)`

    by inserting an instance of `interpret_is_car` with `?a` bound to `car1`. This leaves

    `believe (audience), plausible_in (?d), car (car1)`

    to be supported.

12. Motivate this action with an instance of the rule `must_interpret`.

13. Support the

    `..., case_frame (?u), ...`

    facts with an instance of `do_utterance`.

14. Support

    `believe (audience), plausible_in (?d), car (car1)`

    with an instance of `believed_fact_plausible`. All the pre-conditions of this action are supported.

The remainder of the plan is shown in figure 7–8. Notice that there are two possible first actions, the two utterance actions whose plausibility is provided by the beliefs of the audience.

Utterance 19b is redundant. It is redundant in two very different ways

1. It is tautological. Any entity referred to as "The blue car" is obviously a car and this fact need not be stated.

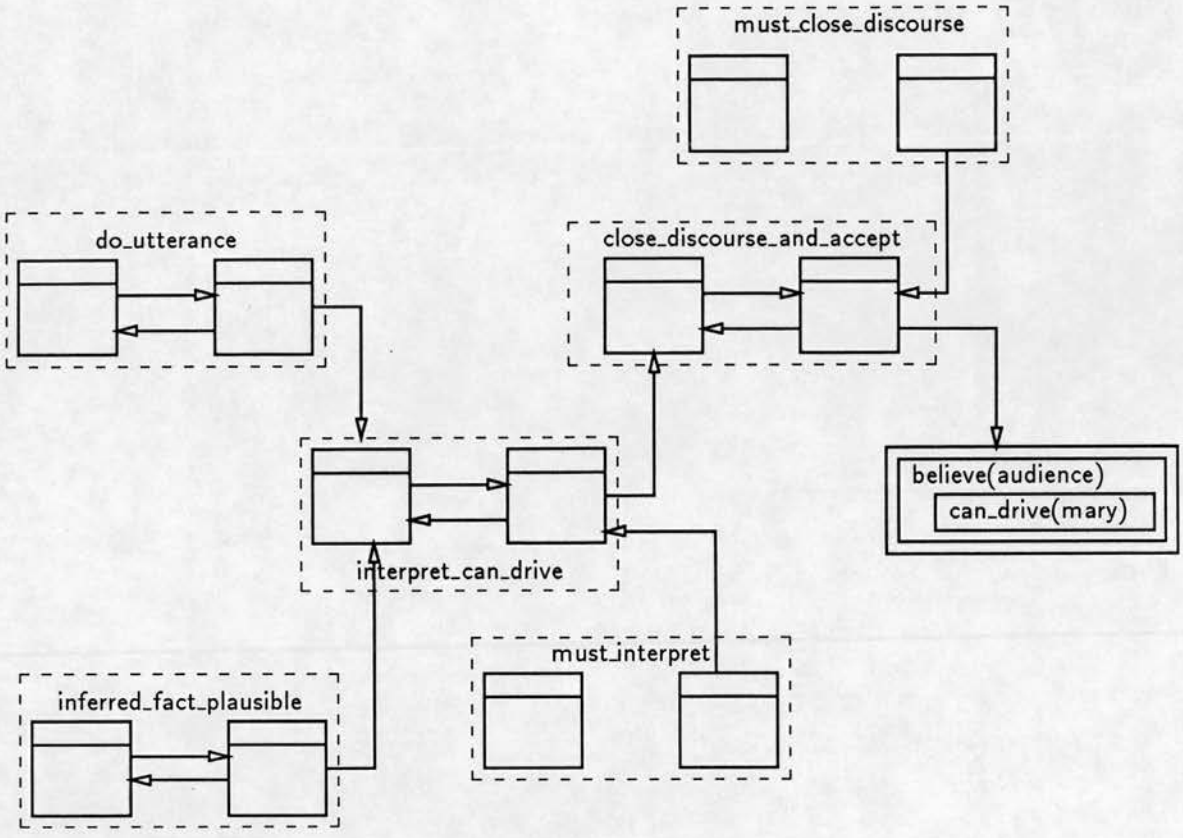


Figure 7–7: The End of the Plan to Guide Inference

2. More subtly, it seems that since the audience *knows* that it is a car, they should be able to bring this fact into the discourse without having to be explicitly stated.

The first of these types of redundancy is the result of our not having included reference evaluation in our interpretation. Since we assume that a "tactical" generation component is producing the surface texts from our case descriptions our system has no way of knowing what information is conveyed as part of the referring expressions in the surface form of an utterance. To avoid this kind of redundancy we might extend any of the interpretation (and so generation) models presented in this chapter along the lines described by [Appelt 1988]. Such an extension, however, lies beyond the scope of the current work.

The second form of redundancy in the example is within the scope of the current work. It is, in fact, the same kind of redundancy as can be found in texts produced by any generating system with a passive, or non-existent, audience model. Our

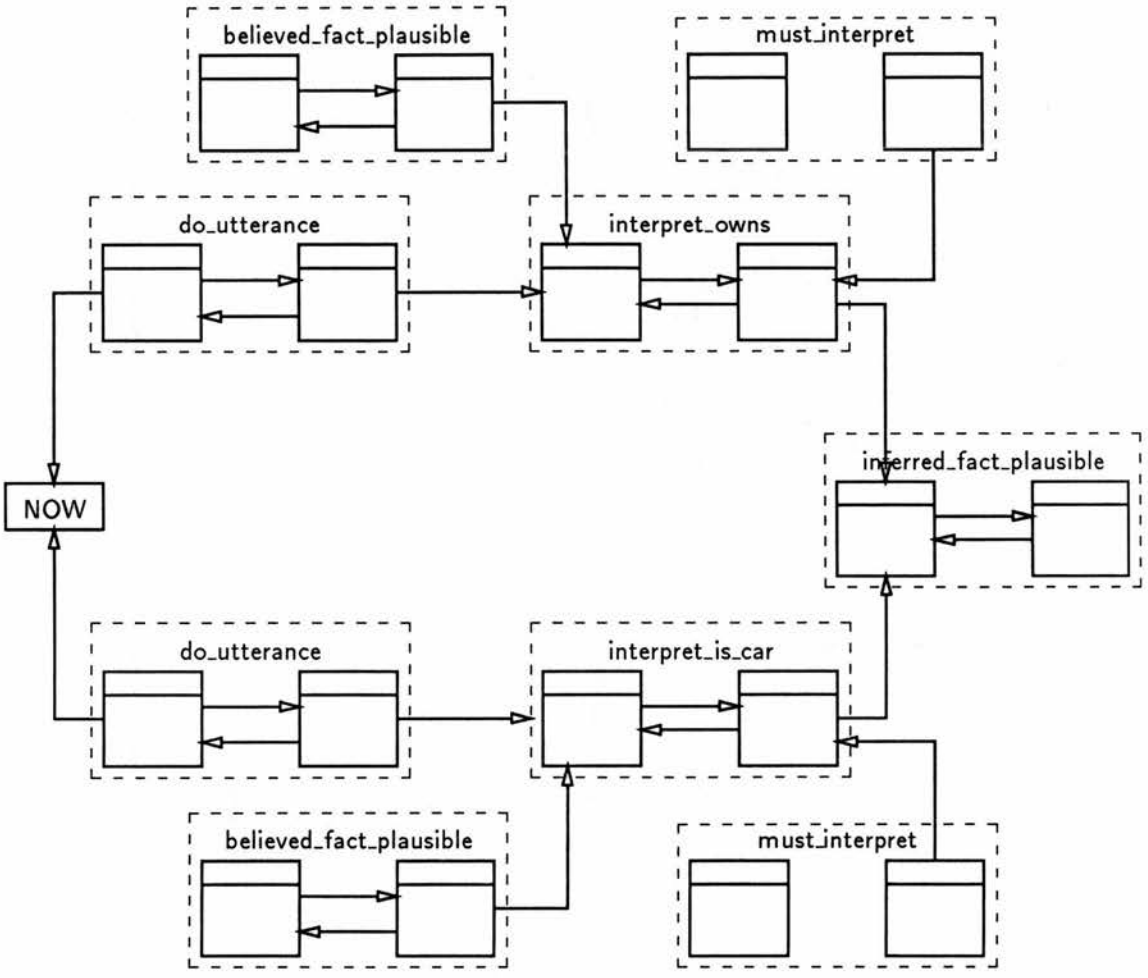


Figure 7–8: Supplying The Preconditions to an Inference

simple interpretation strategy is so closely tied to the behaviour of the speaker that we have, in effect, assumed that the audience will do nothing which is not explicitly demanded by the interpretation of the utterances being made. To remove this kind of redundancy we need make the interpretation strategy more active.

§7 Interpretation as Integration

To model more complex interpretation behaviour we need a more interesting concept of "interpretation" than we have so far used. Previous models have assumed that interpretation consists of determining the meaning of an utterance and then asserting that meaning, as a belief or as part of a discourse context. The model

presented in this section changes this emphasis somewhat. Interpretation is here taken to be the process of integrating a new utterance into a model of the ongoing discourse.

For the purposes of this section the discourse structure maintained by the audience will still be assumed to be a simple world description, represented as the value of a modal term in the audience's beliefs. A more complex discourse structure will be examined in section §6 of chapter 2.

In the new model, "interpreting" an utterance will consist of —

1. Finding the meaning of the utterance. Most utterances will, of course, have more than one meaning; it is the task of the speaker to ensure that the audience chooses the correct meaning.

2. Modifying the discourse context so that this meaning is true. There may be more than one way to do this and once more it is the speaker's task to ensure that the correct modifications are made.

We enforce this interpretation strategy by providing the audience with only one action which can result in an utterance being interpreted.

```
action interpret_utterance
from
    utterance (?u)
    current_discourse (?d)
    meaning (?u) = ?meaning
    discourse_context (?d) = {
        ?meaning
        }
to
    interpreted (ME, ?u)
end
```

This action allows the audience to assert that the utterance has been interpreted when a possible meaning of the utterance is asserted in the current discourse context.

The only kind of modification which we shall allow the audience to make to the discourse context is the assertion of plausible facts, where plausibility is defined as was done in section §6.

```
action assert_plausible_fact
from
    current_discourse (?d)
    plausible_in (?d) = {
        ?plausible
        }
to
    discourse_context (?d) = {
        ?plausible
        }
end
```

This action allows the audience to support a complex fact of the form `discourse_context (?d), TERM` by supporting `plausible_in (?d), TERM`. Informally, the audience may place any plausible fact in the current discourse context. In addition to this rule our model will require the actions `believed_fact_plausible` and `inferred_fact_plausible` defined in section §6 to allow it to determine what is plausible.

Along with these interpretation actions, we will need actions which the audience can perform to extract the meaning of an utterance. These could be similar to the interpretation actions of the simple interpretation model of section §4. However, a more flexible alternative is to have the meanings of case frame formats be described in the beliefs of the agent doing the interpretation. To do this we include the following action in the domain —

```
action extract_meaning
from
    utterance (?u)
    case_frame (?u) = ?cframe
    template (?type) = {
        template_cf = ?cframe
        template_meaning = ?meaning
        }
to
    meaning (?u) = ?meaning
end
```

Then knowledge of how to find the meaning for a case frame can be included in the audience's initial beliefs. For instance, the following additions to the initial state will allow the audience to interpret the three sentence forms in the last example.

```
believe (audience) = {
    template (owns_thing) = {
        [ ?a ?o ]
        template_cf = {
            agent (?a)
            patient (?o)
            predicate (state_owns)
            }
        template_meaning = {
            owns (?a, ?o)
            }
        }

    template (is_kind) = {
        [ ?p ?a ]
        template_cf = {
            agent (?a)
            predicate (tobe)
            property (?p)
            }
        template_meaning = {
            is (?p, ?a)
            }
        }

    template (has_ability) = {
        [ ?a ?o ]
        template_cf = {
            agent (?a)
            predicate (state_can)
            patient (?o)
            }
        template_meaning = {
            can (?a, ?o)
            }
        }
    }
```

This our first usage of the simple quantification mechanism described in section §5.3.3 of chapter 5. The matching process which is used by the planning algorithm will not bind these place holders and so the template can be used to interpret many utterances without becoming instantiated.

In moving to this type of interpretation action we have also made the representation of the state of the world a little more abstract. A the remainder of the initial state of the world for this example is as follows —

```
believe (audience) = {
    owns (mary, car1)
    is (car1, kind_car)
    rule (rule_can_drive) = {
         [ ?car ?person ]
         antecedent = {
             is (?car, kind_car)
             owns (?person, ?car)
             }
         consequent = {
             can (?person, action_drive)
             }
         }
    }
```

If the system is started with this as the 'state of the world' and the complex fact `believe (audience), can (mary, action_drive)`, the behaviour of the system is initially very similar to that of the previous system —

1. Support this unsupported fact with an instance of `close_discourse_ and_accept` performed by the audience and motivate this action by ensuring that an instance of the rule `must_close_discourse` is included in the plan. This will leave the complex fact

   ```
   believe (audience), discourse_context (?d),
   can (mary, action_drive)
   ```

   unsupported.

2. Support this new unsupported fact by inserting an instance of `assert_plausible_fact` with the place holder `?plausible` bound to `can (mary, action_drive)`. This will leave

   ```
   believe (audience), plausible_in (?d), can (mary, action_drive)
   ```

   unsupported

3. The `assert_plausible_fact` action must be motivated. This can be done by assuming that the audience has an instance of `interpret_utterance` in their plan with the complex fact

   ```
   believe (audience), discourse_context (?d),
   can (mary, action_drive)
   ```

   supplying its pre-condition. This means that `?meaning` must be bound to `can (mary, action_drive)`; so the complex fact `believe (audience), meaning (?u), can (mary, action_drive)` will remain unsupported.

This `interpret_utterance` action can be motivated with an instance of `must_interpret` .

4. The complex fact

   `believe (audience), meaning (?u), can (mary, action_drive)`

   can be supported with an instance of `extract_meaning` with the place holder `?type` bound to `has_ability`. This, in turn, will add unsupported complex facts describing the required case frame of the utterance to the plan causing an instance of `do_utterance` to be planned.

When the system's planning has reached this point it will have a plan which is essentially the same as that shown in figure 7–7; the only important difference is the presence of the more general `extract_meaning` action in place of `interpret_can_drive`.

At this point the speaker is left with the task of ensuring that the audience will find `can (mary, action_drive)` "plausible". In the previous model this was achieved by planning utterances which supplied the antecedents of the inference that mary can drive. Since in this model the audience has been given the ability to place plausible facts in the discourse context without their being explicitly stated in the text being interpreted, the speaker need only ensure that all the antecedents of the inference are plausible. In this case, since both of the antecedents are believed by the audience the system can simply insert two instances of `believed_fact_plausible` into its plan, to be performed by the audience.

In addition to eliminating trivial utterances as in the previous example, this audience model is able to detect "side effects" of the interpretation process. Consider the following utterances in a context where speaker and audience both know that the domain of interest is a simple "blocks world" —

20a) Block3 is on Block4.

20b) so, Block4 is not clear.

To interpret this text, the audience would need an inference rule which described the relationship between `on (?a, ?b)` and `clear (?a)`. Assume the following rule is known to the audience –

```
rule (rule_on) = {
    [ ?a ?b ]
    antecedent = {
        is_block (?a)
        on (?a, ?b)
        }
    consequent = {
       ~clear (?a)
        is_block (?b)
        }
    }
```

The audience could use this rule to determine that `~clear (block3)` is plausible in the above text. In doing so the audience would also infer that `block4` is a block and the speaker would know that this inference had been made and so could assume the audience accepts this fact in planning later utterances.

Since the system treats inference as just another action, it can also *plan* for such side effects of inference. For instance given a choice of more than one inference rule which might be used by the audience, the system can choose to guide the audience towards that one which causes "side effects" helpful to the rest of the plan.

Although the audience model described in this section is still far too simple to form the basis of an interesting text planner it does show how the planning process can take account of and guide reasoning by the intended audience of a text. In order to generalise the model we shall need a much more complex notion of 'interpretation', for instance to plan reasonable text it is important that the system have some way of tracking the overall 'focus' of the discourse so that it can distinguish between facts which are available to the audience for performing inferences and those which, although known to the audience, are not so available.

§8 Extracting 'Points'

We will now present a rather more complex model of discourse interpretation, based on that presented in section §6 of chapter 2. In this section we describe how that model can be encoded as a Riple domain. Section §9 presents examples of how this model can be used by Riple to plan texts.

§8.1 Encoding Text Structure in Riple

Recall from section §6.2 of chapter 2 that a discourse structure has 6 parts —

- A description.
- A set of assumptions.
- A set of effects.
- A set of markers.
- A point.
- Information on this context's place in the discourse.

Since we wish to manipulate discourse contexts as we might any other kind of object, the most obvious representation is to have a discourse be represented by an 'entity'. The parts of the context can then be indicated by assigning values to terms built from the entity representing it. We shall use the following functors to encode the structure of a discourse context —

`world_of/1` This is a modal functor. the term `world_of (?context)` has as its value a view representing the description of the context `?context`.

`assumptions_of/1`

`effects_of/1` Similarly these functors are used to construct terms whose values are the assumptions and effects of a context respectively.

`has_marker/2` The boolean term `has_marker (?context, ?marker)` is used to indicate that `?context` has marker `?marker`. The markers we need for the rules presented in section §6.4 of chapter 2 are —

- `m_subjunctive` Placed on contexts representing utterances which are marked as counter factual (for instance the 'if' part of a conditional).
- `m_contrasting` Placed on contexts representing utterances which are marked as contrasting, for instance by the presence of the cue-word "but".
- `m_conclusive` Placed on contexts representing utterances which are marked as having the force of conclusions, for instance by the presence of the cue-word "so".

`point_of/2` The boolean term `point_of (?context, ?point)` is used to indicate that `?context` has the point `?point`. The points used for this simple model are —

- `p_none`
- `p_reminder`
- `p_inference`
- `p_conclusion`
- `p_assumption`
- `p_correction`

These correspond to the types of point described in section §6.2 of chapter 2.

`parent_of/2` The boolean term `parent_of (?context1, ?context2)` will be true iff `?context1` is a child of `?context2`.

The contexts themselves will always be introduced into the plan as unbound variables. New entities will be created to represent the contexts as utterances are made, as described in section §4.1 of chapter 5.

§8.2 Encoding the Point Determination Rules

In order to be able to encode the point determination rules from section §6.4 of chapter 2 we must define the notion of "plausible" which is to be used. We use the same definition of plausibility put forward in section §6, a fact is plausible in a context iff it is either part of that context's world model or if it can be inferred from the description. We must extend the rules slightly to allow the assumptions of the context being examined to be assumed plausible and to allow the audience to make new assumptions.. The actual rules we need are –

```
action plausible_if_in_context

from
    world_of (?context) = {
        ?facts
        }
to
    plausible_in (?context, ?assumpts) = {
        ?facts
        }
end

action plausible_if_in_assumptions

from
    assumptions_of (?assumpts) = {
        ?facts
        }
to
    plausible_in (?context, ?assumpts) = {
        ?facts
        }
end
```

```
action plausible_if_inferable

from
    rule (?rule) = {
        antecedent = ?precons
        consequent = ?facts
        }

    plausible_in (?context, ?assumpts) = {
        ?precons
        }
to
    plausible_in (?context, ?assumpts) = {
        ?facts
        }
    world_of (?context) = {
        ?facts
        }
    effects_of (?context) = {
        ?facts
        }
end

action plausible_if_inferable_in_context

from
    world_of (?context) = {
        rule (?rule) = {
            antecedent = ?precons
            consequent = ?facts
            }

        plausible_in (?context, ?assumpts) = {
            ?precons
            }
        }
to
    plausible_in (?context, ?assumpts) = {
        ?facts
        }
    world_of (?context) = {
        ?facts
        }
    effects_of (?context) = {
        ?facts
        }
end
```

```
action plausible_by_assumption

from
    current_context (?current)
to
    assumptions_of (?current) = {
       ?facts
       }
    plausible_in (?current, ?assumpts) = {
        ?facts
        }
end
```

It is important that `plausible_by_assumption` has a low priority since it is generally not something the speaker wants the audience to do. If the audience just assumes that things are plausible then they will end up with a conditional interpretation of the presented text. The speaker must therefore plan to make one of the other plausibility rules applicable except when making an assumption is safe.

We encode the point determination rules as six actions. The actions are given to the system in the order shown here to give the rules the correct priorities.

```
action determine_no_point

from
    effects_of (?new) = ?effects
    current_context (?current)
    world_of (?current) = {
        ?effect
        }

to
    point_of (?new, p_none)
end

action determine_reminder_point

from
    effects_of (?new) = ?effects
    current_context (?current)
    parent_of (?current, ?surrounding)
    world_of (?surrounding) = {
        ?effect
        }
to
    point_of (?new, p_reminder)
end
```

```
action determine_conclusion_point

from
    current_context (?current)
    has_marker (?new, m_conclusive)
    effects_of (?new) = ?effects
    plausible_in (?current, ?new) = {
        ?effects
        }
to
    point_of (?new, p_conclusion)
end

action determine_inference_point

from
    current_context (?current)
    effects_of (?new) = ?effects
    plausible_in (?current, ?new) = {
        ?effects
        }
to
    point_of (?new, p_inference)
end

action determine_assumption_point

from
    has_mark (?new, m_subjunctive)
to
    point_of (?new, p_assumption)
end

action determine_correction_point

from
    current_context (?current)
    has_mark (?new, m_contrast)
    effects_of (?new) = ?effects
    world_of (?current) = {
      ~ ?effects
        }
to
    point_of (?new, p_correction)
end
```

§8.3 Closing and Interpreting Contexts

The actions to be performed when closing contexts of various types were described in section §6.5 of chapter 2, in this section we will present the Riple actions which correspond to them.

One thing which must be done in these actions is to make the 'point" of the parent context unknown. This is done to force the point to be reconsidered when the parent is closed to take into account the changes made by interpreting its children. In this way we avoid the problem noted in footnote 4 on page 51 of determining the point twice when no change has occurred in the context.

The rules for contexts which have been determined to have point 'assumption', 'correction' or 'reminder' are all very similar —

```
action close_reminder_context

from
    current_context (?context)
    parent_of (?current, ?parent)
    point_of (?current, p_reminder)
    effects_of (?context) = ?effects
    point_of (?parent, ?ppoint)
to
    world_of (?parent) = {
        ?effects
        }
    current_context (?parent)
  ~ point_of (?parent, ?ppoint)
end

action close_assumption_context

from
    current_context (?context)
    parent_of (?context, ?parent)
    point_of (?context, p_assumption)
    effects_of (?context) = ?effects
    point_of (?parent, ?ppoint)
to
    assumptions_of (?parent) = {
        ?effects
        }
    current_context (?parent)
  ~ point_of (?parent, ?ppoint)
end
```

```
action close_correction_context

from
    current_context (?context)
    parent_of (?current, ?parent)
    point_of (?current, p_correction)
    effects_of (?context) = ?effects
    point_of (?parent, ?ppoint)
to
    world_of (?parent) = {
        ?effects
        }
    current_context (?parent)
  ~ point_of (?parent, ?ppoint)
end
```

The actions for closing contexts with point 'conclusion' and for those with point 'inference' are also parallel. Each 'point' has two associated actions which are selected depending on whether the assumptions of the context being closed are plausible of not.

```
action close_inference_context

from
    current_context (?context)
    parent_of (?context, ?parent)
    point_of (?context, p_inference)
    effects_of (?context) = ?effects
    assumptions_of (?context) = ?assumptions
    plausible_in (?parent, nil) = {
        ?assumptions
        }
    point_of (?parent, ?ppoint)
to
    world_of (?parent) = {
        ?effects
        }
    current_context (?parent)
  ~ point_of (?parent, ?ppoint)
end
```

```
action close_inference_context_conditional

from
    current_context (?context)
    parent_of (?context, ?parent)
    point_of (?context, p_inference)
    effects_of (?context) = ?effects
    assumptions_of (?context) = ?assumptions
    point_of (?parent, ?ppoint)
to
    world_of (?parent) = {
        rule (?rule) = {
            antecedent = ?assumptions
            consequent = ?effects
            }
    current_context (?parent)
  ~ point_of (?parent, ?ppoint)
end

action close_conclusion_context

from
    current_context (?context)
    parent_of (?context, ?parent)
    point_of (?context, p_conclusion)
    effects_of (?context) = ?effects
    assumptions_of (?context) = ?assumptions
    plausible_in (?parent, nil) = {
        ?assumptions
        }
    point_of (?parent, ?ppoint)
to
    effects_of (?parent) = {
        ?effects
        }
    world_of (?parent) = {
        ?effects
        }
    current_context (?parent)
  ~ point_of (?parent, ?ppoint)
end
```

```
action close_conclusion_context_conditional

from
    current_context (?context)
    parent_of (?context, ?parent)
    point_of (?context, p_conclusion)
    effects_of (?context) = ?effects
    assumptions_of (?context) = ?assumptions
    point_of (?parent, ?ppoint)
to
    effects_of (?parent) = {
        rule (?rule) = {
            antecedent = ?assumptions
            consequent = ?effects
        }
    world_of (?parent) = {
        rule (?rule) = {
            antecedent = ?assumptions
            consequent = ?effects
            }
    current_context (?parent)
  ~ point_of (?parent, ?ppoint)
end
```

One limitation of these rules is that when an inference rule is created, no quantification is done. This means that these rules are fine for uses like "If Kate is tall, she can reach the top shelf" where the speaker introduces an inference rule for a specific use. However uses like "If someone is tall, they can reach a high shelf", where we would wish to create a rule with quantified variables, are outside the scope of the model.

§8.4 The Interpretation Process

We now have actions which will perform all of the operations which form part of the interpretation process. All that is missing is a reason for an agent to perform these actions. In this section we describe how the interpretation process in controlled.

The basic motivation of the audience in this model, as in earlier ones, is to interpret every utterance which is made with them as the intended audience. Thus the audience is given the following rule to govern its behaviour —

```
rule must_interpret
    addressing (?speaker, ME)
    uttered (?speaker, ?utterance)
gives
    interpreted (?utterance)
end
```

Next we must say what it means for an utterance to have been interpreted. We do this by giving the audience an action as follows.

```
action have_interpreted

from
    current_context (?new)
    context_of (?utterance, ?new)
to
    interpreted (?utterance)
end
```

Which says that an utterance has been interpreted when a context has been built for it and that context has been made the current context. The audience must, therefore, have an action which will allow it to make a context current. It can only do this if it can find a point for it. The action is —

```
action make_current

from
    point_of (?new, ?point)
    current_context (?current)
    parent_of (?new, none)
to
    parent_of (?new, ?current)
    current_context (?new)
end
```

Notice that we must demand that the context has no parent since otherwise it would be possible to apply this action multiple times.

How the audience constructs a discourse context from an utterance description is not really important for our purposes here. For simplicity we shall keep the same simple meaning extraction scheme as was used in the model of section §7. The basic utterance 'parsing' action will be —

```
action extract_meaning
from
    utterance (?u)
    case_frame (?u) = ?cframe
    template (?type) = {
        template_cf = ?cframe
        template_meaning = ?meaning
        }
to
    context_of (?utterance, ?new)
    world_of (?new) = ?meaning
    world_of (?new) = ?meaning
    parent_of (?new, none)
end
```

And the audience's knowledge will include interpretation templates for various kinds of utterance. For instance —

```
believe (audience) = {
    template (owns_thing) = {
        [ ?a ?p ]
        template_cf = {
            agent (?a)
            patient (?p)
            predicate (owns_predicate)
            }
        template_meaning = {
            owns (?a, ?p)
            }
        }
    }
```

Obviously this would have to be elaborated considerably to even begin to describe real utterances, however for our purposes here we need only assume that some actions available to the audience will perform this kind of mapping from utterance description to discourse context. In a more realistic system these 'input processes' might consist of a standard parsing—semantics—reference determination system to be driven 'in reverse' as we are here driving the interpretation process or they might consist of a special purpose tactical generation module such as MUMBLE [McDonald 1981] or Penman [Mann and Matthiessen 1983].

In addition there needs to be a way for features of the utterance to be reflected in marking of the discourse context. For this we may assume actions of the following form —

```
action mark_contrasting

from
    case_frame (?utterance) = {
       has_cue_word (?utterance, cw_but)
       }
    context_of (?utterance, ?context)
to
    has_mark (?context, m_contrast)
end
```

Once again, the precise details are unimportant, all that we shall assume is that the audience has a well defined set of actions which allow it to move from states where it has a description of an utterance to states where it knows that the discourse context associated with that utterance has certain marks.

Finally we must make sure that the beginning and end of a discourse are treated correctly. At the beginning of a discourse there will be no 'current' context. The following action allows the audience to assume that a discourse has started when they need to know the current discourse context.

```
action start_discourse

from
    current_context (none)
to
   current_context (?context)
   parent_of (?context, none)
end
```

The end of a discourse must cause all of the remaining open discourse contexts to be closed. The following rule will make sure this happens.

```
rule must_close_context
    parent_of (?context, ?parent)
    current_context (?context)
gives
    current_context (?parent)
end
```

This rule simply ensures that if we give a context a child, that context will become the current context again at a later date. The only way this can happen is if the child is closed. A special case of this rule occurs when `?parent` is `none`, that is when closing the top level context. There are two obvious approaches to this, either `none` could be set up to be pseudo-context with world model the audience's beliefs

etc, leading to an audience who takes texts fairly literally, or a specialised set of context closing rules could be defined for the top level discourse context describing the effects of various types of texts, for instance reminders from different speakers might have different effects. For present purposes these choices are not important since we are interested in how planning and text structure combine and not in social issues such as how to react when someone tells you something. In the examples below we shall assume that the fragments being discussed take place in some wider context, whether that is a larger discourse or rules for interpreting a top level context.

§9 Planning Structured Texts using 'Points'

In this section we will show how the audience model formalised in section §8 allows Riple to plan texts whose structure is important to their meaning. The texts which the system will plan here are the ones which were used as examples of interpretation in section §6.7 of chapter 2.

In the following sections we will omit many of the details of the planning process. Hopefully the earlier examples have given the reader a good idea of how the planning process works; here we are concerned with how planning concepts such as 'motivation', conflict detection and support interact with linguistic concepts such as text structure and cue words.

It should be emphasised again that the purpose of this model is not to produce linguistically interesting descriptions of text structure but to explore the interaction between audience modelling and text planning choices.

§9.1 Planning a Reminder

The first fragment from section §6.7 of chapter 2 is fairly trivial and we present it only so that the general form of the text planning process can be seen in some detail. In the later examples we shall abstract away from this detail.

We assume that in the course of planning some larger text, the system has to support the goal of having the fact `owns (mary, car45)` in the description of the current discourse context. This might be needed to allow the audience to make an

inference without having to make it an assumption for instance. In this situation the system would have an unsupported complex fact

```
believe (audience), world_model (c123), owns (mary, car45)
```

where `c123` is believed by the audience to be the current context.

The system would proceed as follows —

1. To support this fact it would include an instance of the action `close_reminder_context` in the plan. This would leave unsupported the preconditions that there be an open context, that that context have point "reminder" and that it's effects contain `owns (mary, car45)`.

2. This action can be trivially motivated, there is an open context so the rule `close_all_contexts` will cause it to need to be closed. It is also the only action which will work for a context with point "reminder".

3. To support the precondition that the open context have point reminder, the system will need to include an instance of the action `determine_reminder_point` in the plan. This will have precondition that `owns (mary, car45)` be true in the description of the surrounding context. We can assume this is true for the purposes of this example.

4. To support the fact that the open context have effect `owns (mary, car45)` the simplest option is to plan to make an utterance which will be assigned a context of this type by the lower level processes. We represent this here by having the system plan an action of the type `extract_meaning` with the `?meaning` place holder bound to a view containing `owns (mary, car45)`. The only possible way of supporting the preconditions of this action will be to bind the `?type` place holder to `owns_thing`. This will in turn force the case frame associated with the utterance to have the correct form.

   This `extract_meaning` action would also cause there to be an open context, so supplying another precondition, the identity of this context will remain undefined (i.e. an unbound place holder) until the action is performed.

Thus the correct utterance (or one with similar effect) would be included in the plan. The remaining unsupported preconditions which the above actions would leave in the plan would be trivial ones (such as the speaker must be addressing the audience) and so the plan is complete.

The following examples are more complex, but generally the planning of each utterance has the form above, plan to have a "close" action performed to get some effect, this places restrictions on the open context and those restrictions are fulfilled by a combination of a point determination action, which places more restrictions on the context, and either an utterance action or, for more complex examples, a number of subsidiary "close" actions, eventually terminating in expansion to utterance actions.

§9.2 Planning for an Inference

We will now see how the fragment of discourse presented in section §6.7.2 of chapter 2 might be planned. The aim here is to see how structure is introduced into the plan by the need to alter the effects of a context already planned.

We assume for the present example that the speaker's plan contains an unsupported precondition that the effects of the current context include the fact that John is a good choice for the team. We must also assume that the audience has available the inference rules required to deduce from John being a police officer the plausibility of his being tall and from his being tall the plausibility of his being a good choice for a basketball team and, of course, that John being a police officer is already in the current context.

Planning proceeds as follows —

1. To make a fact true in the effects of a context the only actions available are the context closing actions. In fact only a conclusion will modify the effects, so an instance of `close_conclusion_context` will be planned to be performed by the audience with the effects of the context to be closed including `good_for_team (?john)`.

2. The `point_of` precondition of this new action will be supported in the only way possible, by an instance of `determine_conclusion_point`, also to be performed by the audience.

3. Most of the remaining unsupported preconditions are simply constraints on the new context which can be satisfied by an utterance action of the correct form. However, the `determine_conclusion_point` action will leave as a precondition the plausibility of `good_for_team (?john)`. If we assume that

this fact is in neither the assumptions nor the world model of the current context and that there is no simple way to make them true, the only action which can be used to support this plausibility is `plausible_if_inferable` and so an instance of this will be planned. This will give rise to a new unsupported precondition, the plausibility of John's being tall.

4. The plausibility of John's being tall can be supported by having it be in the description of the current context and this in turn can be supported by having the audience close a context with point `p_inference` and the correct effects.

5. So, now the speaker must force this new context's point to be `p_inference`. Once again it is necessary to support the plausibility of its effects, in this case `is_tall (john)`, and once again `plausible_if_inferable` must be used. In this case the inference rule used will be that if John is a police officer then he is tall.

6. Now, the plausibility of John being a police officer *can* be supported without invoking `plausible_if_inferable` again. It can be safely made part of the assumptions of the context being planned since it's plausibility in the surrounding context can be trivially deduced[4].

7. Finally the precondition that `is_police_officer (john)` is an assumption must be supported. This can be done using `close_assumption_context` which in turn must have its preconditions supported by an `determine_assumption_point` and together these will give rise to an utterance action which will describe an utterance of the form "If John is a police officer".

One thing which is obvious from this example is how critical the priorities of the various actions available to the planner is to the form of the text. For instance the fact that the context closing actions are given a higher precedence than the plausibility rules ensures that the system acting as a speaker will prefer plans where each inference by the audience is guided by an explicit utterance by the speaker. Reversing these priorities would make the system prefer to plan the text

---

[4]Here we gloss over the search necessary to find that this is an acceptable plan whereas, for instance, making `is_tall (john)` an assumption is not.

"John is a police officer, so he is a good choice for the basketball team"

which leaves the audience to find the inferences to bridge the gap. Similarly a perfectly valid plan would be to introduce a new inference rule to the audience, supply its antecedents and then draw the conclusion, giving a text of the form[5] —

"If John is a police officer he is tall, so he is a good choice for the basket ball team. He is a police officer, so he is a good choice."

Where the utterance "He is a police officer" could not be interpreted inside the context of the earlier utterances causing them to be closed using `close_inference_context_conditional` giving rise to an inference rule which is used to interpret the later utterances.

The choice between such different possible text structures is one of style and lies beyond the scope of this work. A system which was to be able to choose its style intelligently rather than having it 'wired in' would need a much more sophisticated model of its audience and the (social) context in which the discourse being planned would take place.

§9.3 Cancelling a Plausible Inference

One way in which an audience model is useful in text planning is in allowing the speaker to anticipate and correct for situations in which the audience might be expected to incorrectly infer something based on the text. Consider the simple text —

"Kate is a police officer but she isn't very tall."

This might be planned either if it was important for a later part of the text that the audience not infer that Kate was tall (for instance if such an assumption would block a required interpretation) or if there was a non-linguistic need for the audience's knowledge of Kate to be correct (they might need to recognise her for instance, in a system capable of dealing with more subtle social constraints on behaviour it might be important that the speaker not be seen to mislead the

---

[5]Once again this is rather strange due to the limitations of having everything be present tense

audience). This would result in either and unsupported fact later in the plan or, if that fact had been supported, a constraint which would prevent a discourse context containing the unwanted fact from being incorporated into it's parent. In either case the result would be that planning would be blocked unless it was possible to negate the unwarranted inference. So, for this example we will assume that the plan contains unsupported facts

```
believe (audience), world_of (c456), police_officer (kate)
believe (audience), world_of (c456), ¬ tall (kate)
```

where `c456` is the current context at that point.

A second issue is how the audience comes to make the unwanted inference. Since details of knowledge representation, stereotypes and so on are not of direct interest here we will just assume the following very simple inference rule is believed by the audience —

```
rule (can_be_police_officer) = {
    [?person]
    antecedent = {
        person (?person)
        }
    consequent = {
        police_officer (?person)
        tall (?person)
        has_uniform (?person)
        }
    }
```

This kind of rule represents a stereotype. It's presence in the audiences beliefs indicates both a set of assumptions about a generic police officer and also a willingness to believe that someone is a police officer without further evidence. A more unusual occupation might have a rule with more precondition.

In a more realistic system the connection between being a police officer and being tall might be more indirect, for instance being tall might be a precondition of the plausibility of someone being a police officer and the plausibility of being tall might be based on other assumptions[6].

---

[6]Note that this would mean that this would mean that the speaker's correction would remove a basis for the plausibility of something already inferred. In a keen system this

A final assumption we make is that the system in the course of its operation the system decides to support the `police_officer` term before the `tall` term. This assumption is made for explanatory purposes. Should the system tackle them in the opposite order the planning process would involve a large amount of backtracking to reach a stable plan. The problem of which unsupported term to try and support first to avoid backtracking is in general very difficult and not tackled in this work.

We can now describe the behaviour of the system. The first few actions are similar to previous example.

1. To support `...police_officer (kate)` the system inserts an instance of `close_inference_context` to be performed by the audience.

2. The `point_of` precondition of the close action will be supported by an instance of `determine_inference_point`. This will leave unsupported the plausibility of `...police_officer (kate)`.

3. An utterance action by the speaker provides support for many of the remaining preconditions. What remains to be supported is the plausibility of `...police_officer (kate)`. As in section §9.2 the only way to get this plausibility is with an instance of `plausible_if_inferable`. However in this case the precondition of the inference rule (that kate be a person) is easy to deal with. Since it can be assumed to be part of some surrounding context or in the audience's beliefs the system can allow the audience to use `plausible_by_assumption` since when the current context is closed the assumption will be plausible or at worst can be made plausible by assumption again and so on up the tree.

4. We will now be left with a situation where there are two facts in the effects of the current context, that Kate is a police officer and that she is tall. To close

---

might mean finding a new basis for plausibility. However since the current model does not demand that the plausibility argument for an inference *remains* true, only that one can be found, this model would not force such a rethink. Whether this is reasonable would depend on the type of discourse being modelled. In a casual conversation an audience might well let it pass, while when reading a formal proof they would almost certainly go back and check.

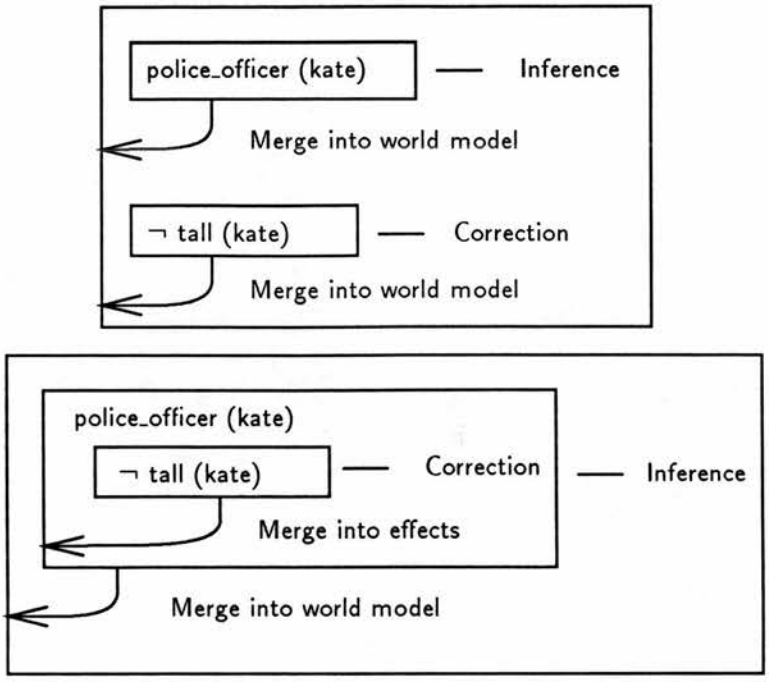


Figure 7–9: Two Alternate Text structures for the Example

the context, the system will insert an instance of `close_inference_context` which will insert these two facts into the world model.

5. Now the system will try and support the fact `believe (audience), world_of (c456), ¬ tall (kate)`. The only way to do this is to close a correction context (there is no way of inferring it and it is certainly not a reminder).

6. The preconditions of the close action can be provided by an instance of `determine_correction_point` and an utterance action. The fact that the correction needs the fact to be corrected to be in the parent context's world model will ensure that the two utterances will be ordered correctly.

It is interesting to consider a variant on this example. If we were to modify `close_correction_context` to place it's effects in its parent's *effects* rather than its parent's world model, the system could not plan as above. Instead it would need to place the correction close (and hence utterance) between the first utterance and the closure of its context. The result would be to change the discourse structure planned as shown in figure 7–9. This change obviously causes the correction to be planned as more closely related to the original statement. The affect on the final

text might be to make it be generated as a non restrictive relative clause rather than as a conjoined sentence, for instance —

"Kate is a police officer who is not tall"

Of course the actual realisation would depend on the surface generation system used.

§9.4 Planning a Longer Discourse

As a final example we now describe how a slightly longer text with more structure could be planned using the mechanisms we have presented. Again this example was first presented in chapter 2.

For this example we need two new inference rules —

```
rule(shopping) = { [?person]
    antecedent = {
        in (?person, shop)        ;;; someone in a shop
        has_money (?person)       ;;; with money
        }
    consequent = {
        can_buy_food (?person)    ;;; can buy food
        }
    }

rule(wallet) = { [?person]
    antecedent = {
        has_wallet(?person)       ;;; someone with their wallet
        }
    consequent = {
        has_money (?person)       ;;; has money
        }
    }
```

We use a constant shop here rather than a variable with property shop just to eliminate an uninteresting step in the planning process.

We assume that we start planning from a situation where the planner has an unsupported term in the plan that the world model of the current discourse context assert that John can buy food. We also assume that the facts that John is in the shop and that he has his wallet are in the current discourse context's world model. We also assume as usual that the necessary case frame interpretation rules are available.

Planning proceeds as follows —

1. The first step is simple, we wish to change the world model so we plan to have the audience close a conclusion context. Then we support that by having them identify it as a conclusion.

2. Now we have to support the plausibility of `can_buy_food (john)` in that context. This can be done by having the audience use the inference rule `shopping` above. We then have to recursively support `in (john, shop)` and `has_money (john)`.

3. An utterance of the correct form to support the effects of the conclusion context can now be planned.

4. `has_money (john)` can be supported by having the audience close a conclusion context.

5. To ensure that this is identified as a conclusion we have to support the plausibility of `has_money (john)`. This can be supported using the inference rule `wallet`. This leaves the plausibility of `has_wallet(?person)` to be supported.

6. `has_wallet(?person)` can be supported by a reminder.

7. `in (john, shop)` can also be supported by a reminder.

8. The remaining unsupported facts can be supported by planning relevant utterances.

This process results in a simple flat discourse structure.

21a) John is in the shop.

21b) He has his wallet.

21c) So he has money.

21d) So he can buy some food.

However if the priorities given to the planning algorithm to different chore types, fix types and actions are changed slightly a different sequence of actions is planned.

1. Again we plan to have the audience close a conclusion context and so have to support the plausibility of `can_buy_food (john)`.

2. We plan to use `plausible_if_inferable_in_context` rather than `plausible_if_inferable` This means the existence of a suitable rule will be unsupported.

3. We can support the existence of the rule by by planning to have a context closed using `close_conclusion_context_conditional`. And ensuring that the effects of the context contains `can_buy_food (john)`.

4. In that context, we can support the effects by planning the closing of a conclusive context. This in turn can be supported by an utterance and the proof of the plausibility of `can_buy_food (john)` in *that* context.

5. This time we choose to use `plausible_if_inferable`. The plausibility of the preconditions of the rule `shopping` now needs to be proven.

6. There is a choice as to how to do this. We could allow the audience to just assume them or we could plan an explicit assumption utterance. For the purposes of the example, we do one of each. `in (john, shop)` is supported with `plausible_by_assumption` while `has_money (john)` is supported by an instance of `close_assumption_context` and `plausible_if_in_assumptions`.

7. After planning utterances to support the correct closing of those contexts what remains is to support `has_money (john)`. For simplicity, we assume this is done as before.

Resulting text might be rendered as —

22a) John is in the shop.

22b) If he has money

22c) then he can buy some food.

22d) He has his wallet.

22e) So he has money.

22f) So he can buy some.

which is the text we interpreted in section §6.7.4 of chapter 2.

The main question raised by this example is, how should a text planner make these low level choices which influence text style. Some of the choices are forced by the audience model, but others are left open.

One solution would be to create a more detailed audience model, for instance if the audience were only able to prove plausibility using inference rules with a single non-assumed term in the antecedent then the speaker would *have to* plan the construction of such rules a step at a time as in this example, resulting in a very formal style of text.

An alternative would be to introduce some form of discourse grammar or schema rules into the system to be used when there is a choice which is not influenced by the audience model. Thus getting some of the best of both worlds. A similar approach could be taken to introduce stylistic rules such as "avoid repeating structures".

§10 Conclusions

In this chapter we have presented a number of possible audience models increasing in complexity from a trivial model which simply accepts the 'meaning' of the speaker's utterance as true, through models including increasing levels of autonomy ending with a relatively complex model capable of tracking complex discourse structures and interpreting the speakers utterances in the context of such a structure.

In all of these systems the audience model has provided the framework within which text planning is performed. In no case has the presence of the audience model proved an obstacle which the speaker has had to overcome, rather the audience model has provides guidance to the text planning system replacing to a great extent the schemas and text structure rules of other text planning systems. The constraints on the system arise from the limited array of options provided by the text planner's linguistic and interpersonal competence (very poor in the models presented here) and that which it assumes of its audience. This kind of 'means vs. ends' constraint is the type which planning systems have been developed to reason about, and because of this the combination of a planning algorithm with an

ability to reason about the audience's interpretive behaviour gives a very natural formulation of text planning.

The style of a text can be changed by changing the priorities of actions available to the audience model. In cases where the audience model does not totally constrain the text the remaining choices have been made arbitrarily, or by human intervention to illustrate a point. This would seem to be a good place to introduce other knowledge into the system, for instance a knowledge of conventional text structures or stylistic rules such as avoiding repetition.

Chapter 8

Discussion

§1 Introduction

In the preceding chapter we described several strategies for interpreting utterances in a discourse and showed how these might be formalised within the framework provided by Riple to be used as audience models in text planning. In this chapter we discuss how well Riple achieves the aims set in chapter 1 —

- Taking the intended audience's reasoning into account in planning a discourse but avoiding the problems of logical omniscience.
- Using the audience model as an aid to discourse planning rather than a constraint.
- Avoiding unnecessarily limiting assumptions.

In section §2 we evaluate Riple according to these criteria, comparing it to previous work on text planning. Section §3 discusses the limitations of Riple and makes some suggestions for how they might be removed.

§2 Evaluation of Riple

In order to plan reasonable discourse an agent must be able to take into account not just the beliefs and desires of the intended audience of the discourse, but also the inferences and assumptions which that audience will make while interpreting the discourse. Previous discourse planning systems have ranged from those with very simple audience models, such as McKeown's TEXT system ([McKeown 1982]), where the only activity assumed by the audience model is the tracking of focus, to systems which use an inference engine to model the behaviour of the audience, such as Appelt's KAMP system ([Appelt 1982]).

Schemata based systems have a large advantage in being able to imitate human produced texts as closely as the implementer wishes. However they do not degrade gracefully, when given a task which is outside their repertoire they have no ability to improvise. By breaking the analysed text up into smaller pieces and using heuristics such as focus tracking the texts can be made more sensitive to context, but they are still only as good as the mechanism they use for selecting which schema to use when for which piece of text. Thus the use of schemata delays the need to be able to reason about the purpose of the text being produced, but does not eliminate it.

On the other hand, if too much general inference power is built into the language/audience model, it becomes hard for the text planning system to predict the effects of its actions.

Riple is an attempt to steer a course between these two extremes, providing the ability to model complex audience behaviour without being subject to the problems of unpredictability and logical omniscience which occur in systems containing an inference engine capable of full first order reasoning.

Riple achieves this balance by explicitly modelling the inference processes of its audience as part of its planning process rather than making inference the responsibility of a separate engine. In doing this we are making the assumption that the inference processes of an audience interpreting a discourse are essentially goal directed rather than taking the form of arbitrary deduction based on the information presented in the discourse.

As an example of this difference, consider the following fragment of discourse —

23a) A prime number is a natural number whose only factors are 1 and itself.

23b) So, 5, 7, 11 and 13 are all prime numbers.

This seems a perfectly reasonable way to introduce the concept of prime numbers to someone who did not know the term. If we ignore the problem of deciding that an abstract description and a number of concrete examples forms a good description of a mathematical concept such as "prime number", an interesting and complex problem in its own right (for instance, see [McKeown et al. 1985]) but one beyond the scope of this work, a speaker planning to explain primeness will still have the problem of deciding how best to convey the description and the examples and the relationship between them.

A system which assumed logical omniscience of its audience would never plan a discourse such as that shown above, since 23b follows from 23a and so the system would see no reason to say it, assuming that the audience would have inferred it.

Riple, on the other hand, assumes that inference is performed only when it aids in the interpretation of the text and would not assume that the audience inferred 23b from 23a without prompting and so would know that it is reasonable to point the inference out to its audience. For example, see the discussion of the effects of changing the salience of different types of inference in section §9.2 of chapter 7.

Another area where modelling the audience's inference process explicitly avoids problems is in deciding what effects a given utterance will have in a context and, symmetrically, what utterance to perform to have a given effect. A system which must plan discourse using a model of its audience which contains a reasoning system capable of first order inference cannot, in general, predict what the effects of introducing a given belief into its audience's view of the world will be, all it can assume is that the resulting audience model state will be consistent. For the same reason it cannot determine which beliefs should be introduced to have a given effect.

Simple discourse planning systems often assume that the effect of an utterance of a given form is fairly predictable from its structure and that its broader effects will be limited to causing the audience to infer a predictable set of 'obvious' conclusions, for example [Mann and Moore 1981] and. However this kind of system is forever trapped between the problems of ensuring that all 'obvious' inferences are planned for and yet not making so many assumptions as to make the text incoherent. In fact, the KDS system described in [Mann and Moore 1981] attempts to walk

this line by allowing the process which assembles paragraphs from fragments to reintroduce information deleted by the 'obviousness' filter. This stage of KDS in many ways performs the task we are advocating in this thesis should be performed by the audience model — controlling the discourse planning task – and with its heuristic measures of text coherence it can be seen as a very simple audience model, albeit one whose behaviour is tied very firmly to one domain. The obvious problem with this approach is that it is hard to see how one could generalise it to a wider domain without the number of heuristics and their interactions becoming unmanageable.

At the other extreme KAMP has such a complex inference engine built into its model of the world that the planning system cannot determine by looking at the stated effects of an action what the wider effects on its audience will be. To avoid this problem Appelt supplies KAMP with schema which describe the effects which the actions can be expected to have in most cases. These schemata are used to guide what would otherwise be a blind search through all possible plans for one which has the desired effects. When the schemata fail to predict the behaviour of an action correctly, KAMP has no way of knowing what actually caused the failure (for example an existing belief of the audience or a side effect of a previous utterance) and so has no way to intelligently decide how to fix its plan.

Riple actions have known effects since they describe only the direct effects of performing the action. Any side effects caused by the audience's (presumed) behaviour while interpreting the utterance will be explicitly included in the plan by the planning algorithm and so further planning will be able to take them into account or correct for them.

In section §9.3 of chapter 7 we showed how Riple can plan discourse taking into account side effects of interpretation. In doing this it uses the audience model, as was suggested in chapter 1 as a source of suggestions for the planning process and a way of choosing between what might otherwise seem very similar alternatives. The planning system uses the audience model as a resource rather than as a constraint.

§3 Limitations

The model of text planning developed in this thesis is based on the assumption that language use is not qualitatively different from other types of action involving more than one agent. This assumption led to the decision to embed the text planning system in a more general model of planned action, the planning system Riple. However, planning in a multi-agent environment, cooperation between independent agents and representing the beliefs of others are significant research problems in their own right which lie beyond the scope of this work. For this reason, the model of planned action represented by Riple contains a number of assumptions and simplifications which were chosen to avoid the harder problems of multi-agent planning, as was described in chapter 5 and discussed further below.

These assumptions and simplifications give rise to a number of limitations, both in the model of planed action and in the model of text planning. In this section we examine some of these limitations.

§3.1 Limitations Arising out of the Simplifying Assumptions

The most important assumptions are those associated with the planning model built into Riple and described in section §2 of chapter 5. And of these the most restrictive seems to be the assumption which was called "Agent Similarity" —

Agent Similarity All the agents who exist in the universe are assumed to be of a similar kind.

This assumption has the effect of limiting Riple's audience model to be built of the simple building blocks available in the planning system itself. Although this is a rather large restriction — there are many heuristics about discourse interpretation which we might wish to incorporate into our audience model without being able to express them within the relatively impoverished notation available in Riple — it should be clear from the preceding discussion that the advantages of having all of the behaviour of the audience model be available to the planning process more than outweigh the difficulties. Section §3 below discusses the limitations of the system as it now stands and proposes directions for further investigation.

After "Agent Similarity", the most restrictive assumptions in Riple are those which form part of the basis of AI planning systems in general. Almost all work in planning assumes that actions are discrete, that their results are predictable and known and that the universe can be described in a modular way. These assumptions carry over into Riple leading to assumptions of discreteness in agents beliefs (for instance a Riple agent might believe that a certain car is blue, but not that it has a particular shade expressed on some kind of continuous scale). Also it is a basic assumptions of Riple's underlying planning model that events in the universe are instantaneous and so cannot overlap. This latter assumption is likely to be much more restrictive should an attempt be made to plan dialogue rather than single speaker discourse with Riple. Phenomena such as turn taking and interruptions would be impossible to model without removing these basic assumptions of discreteness.

§3.2 Limitations in the Representation

Moore ([Moore 1981]) describes a number of problems which seem unavoidable in representations which explicitly describe the beliefs of other agents (or indeed, other 'modal' concepts). It is not possible in this kind of model to represent some kinds of disjunctive beliefs and/or beliefs about disjunctive beliefs; for example consider the utterances 24 and 25 –

24) John believes that either Mary or Kate has the book.

25) Either John believes that Mary has the book or he believes that Kate has the book.

To distinguish these two, obviously very different, statements, we must be able to represent disjunction "inside" or "outside" a modal context.

Moore suggests a form of representation based on "possible worlds" which does allow for this kind of distinction; however, as was discussed in section §3.3 of chapter 4, Appelt's use of this indirect representation in KAMP was the main cause of the difficulty which KAMP faced planning for the actions of it's audience.

In the representation used in Riple (described in chapter 5 and defined more formally in chapter A) this issue is avoided, rather than solved, since disjunction cannot be represented at all. Obviously further work in this area is needed.

The very simple type of term which forms the basis of the representation also places limitations of the expressiveness on the model. There is no way in this representation to describe continuous phenomena or the kind of one-to-one relationships usually represented in the predicate calculus by functions.

Finally, the representation has only a limited ability to allow for the manipulation of sets of assumptions. The simple ploy described in section §3.2.4 of chapter B allows for simple wild card style matching of arbitrary terms and their values, but it would be a major increase in the power of the representation should a more general mechanism for representing this kind of information be introduced.

§3.3 Limitations of the Planning Algorithm

The planning algorithm used by Riple is very simple, consisting of just repeatedly attempting to support states of affairs in the plan, execute or motivate actions or fire rules. There are many phenomena in multi-agent planning which this simple model ignores –

- Conditional plans.
- Execution monitoring.
- Assigning responsibility for supporting goals.
- Modelling other agents' planning behaviour.

All of these problems might improve the text planning ability of the system. Conditional plans allow the planner to delay, or even ignore, choices which otherwise might lead to an over-constrained plan.

Execution monitoring would allow the system to recognise communication problems when they arise. Responsibility assignment is done very simply in Riple' by assuming that the agent whose goals are being supported when an action is inserted into the plan is responsible for achieving those goals and having the system work out complete plans of how it thinks the agents will achieve this support. This is enough for the simple interactions modelled in the current work, but will fail on more complex interactions where the system is likely to get bogged down in the minutiae of modelling other agents plans. Modelling the audience's planning

would allow the system to plan for "second order" effects by influencing the way in which the audience eliminates conflicts in their plan.

All of the above are subjects of interest to researches in distributed AI and, hopefully, work in that field may produce results which can be used in future work on discourse planning.

§3.4 Limitations in the Language Model

The interpretation models developed in chapter 7 cover only a small section of the task of interpreting discourse. They were developed in a more or less ad-hoc manner to illustrate the interaction between planning and interpreting discourse and no real attempt was made to construct a linguistically interesting theory.

Only the selection and ordering of the information conveyed and the insertion of cue-words to signal the relationship between utterances was described. However, even in this small domain there are obvious limitations to the model. The inferences modelled by the interpretation strategies of chapter 7 are simple implication and default assumptions. A more complete model would need to be able to represent quantification and disjunction. If the models were to be extended in this way (assuming that a suitable extension to the representation could be defined) then it would also be necessary to have a more formal model of inference as the basis of the interpretation.

In order to focus on the interaction between speaker and audience, the 'points' model was intentionally designed to rely heavily on the developing structure of the text to control what interpretations were made. Because of this it takes only a few markers representing the surface form into account. In reality a human being interpreting human produced text has a great deal of help from the surface form of utterances, for instance intonation and pauses in spoken text or formatting in written text is used to indicate text structure. While this limitation of the model was consciously imposed in order to make it clear that it was the audience model and not surface linguistic constraints which were controlling the planning process, it is clear that a more practical system would need to have more surface linguistic knowledge. Such a system could avoid always having to deal in detail with the inner workings of the audience model, while still having those workings present to provide guidance when difficult decisions are made and to warn when purely grammatical decisions conflict with the requirements of communication.

Probably the most important area of discourse planning which was not tackled is that of planning referring expressions. This problem interacts to such an extent with problems of content selection and ordering that its lack must be seen as a major limitation.

§3.5 Limitations of the Program

The computer program which implements the mechanisms described in this thesis has only one major limitation beyond those which arise from limitations of the abstract model already discussed. The structures which represent views in the program use (partial-) functions to represent the mapping from entities in the view to entities in the enclosing context. An examination of the structures defined in chapter B and the mechanisms of chapter C will show that in all instances the mappings defined will, in fact, be functions in the entities of the view. Although the formal definition of a view in section §2.3 of chapter A is more general, allowing an arbitrary relation as the mapping between entities within and outside the view following Fauconnier [Fauconnier 1985], Riple cannot, at present, take advantage of this additional flexibility.

The discussion of the program at the end of chapter C includes some comments on other choices which were made in implementing the program which introduce limitations. It is clear that further work on more general control strategies would remove some of these, but to what extent this would benefit Riple as a text planner is unclear.

§4 Summary and Further Work

The combination of a well-defined model of planned action and a simple model of the interpretation of texts forms a model of text planning which has a number of advantages over schema or logic based models.

- Explicit modelling of the audience's beliefs and intentions allows much greater flexibility than is possible with schemata.
- Explicit modelling of the audience's inference activity allows the system to directly determine what must be changed to counter an assumption or block an inference.

However the model described here has a number of limitations, the most outstanding of which is the lack of a more complete and well founded model of discourse interpretation to form the audience model of the system. It is probable that in order to develop such a model a number of limitations of the planning system and representation used by Riple would have to be removed. Perhaps the most important such limitations are the lack of disjunction and functions in the representation.

Another area where further work might prove fruitful is the modelling of dialogue phenomena such as turn taking and interruptions. This too would necessitate extensions to the current system; in this case the most limiting factor is the poverty of the model of multi-agent planning which Riple instantiates. Such things as conditional plans and the ability to explicitly model the planning process within the planner would seem to be necessary for the modelling of more complex communicative behaviour.

A final set of interesting questions arise out of the examples in section §9 of chapter 7. It was noted there that changing small details in the formulation of the interpretation model (for instance the priorities of actions or the decision of whether or not a correction affects the 'effects' of a surrounding context) can produce large differences in the 'style' of the planned texts. The relationship between the audience model and stylistic choices might be a good angle from which to approach the construction of more principled audience models. For instance The 'points' model discourages neither very deeply nested text structure, which might be confusing, nor very flat almost unstructured texts of the form "X and so Y and so Z and so ..." which are at best boring and at worst cause human readers to loose track of where the discourse is going. It might be useful to limit the number of open contexts allowed and to introduce some mechanism for having facts be deleted from a context after a while to counter these effects producing more readable texts and a possibly more realistic interpretation model.

Appendix A

A Representation for Multiple World Models

§1 Introduction

This appendix defines a class of structures which can be used to describe states of affairs which require the representation of modal concepts such as the beliefs of multiple agents or temporal sequences of states. This representation is a formalisation of that developed in sections §3 and §4 of chapter 5.

The representations built have the following properties —

- The representation is 'intensional'; that is, the components of the representation of a state describe parts of the state rather than directly corresponding to those parts.
- The representation of a complex state contains as parts representations of other states. Each such representation is an independent description of a state of affairs involving a set of entities unique to that description.
- In putting together a number of such state descriptions to form a larger one the relationships between the entities in each description must be defined explicitly.

In the definitions below indented text in *italics* is commentary, things in **bold** are sets, things with an Initial capital are functions and those in CAPITALS are constants

The notation used is as follows

$F : \mathbf{A} \rightarrow \mathbf{B}$ F is a total function with domain **A** and range **B**.

$F :_{part} \mathbf{A} \rightarrow \mathbf{B}$ F is a partial function with domain a subset of **A** and range **B**.

$\{a_1 \rightarrow v_1, a_2 \rightarrow v_2 \ldots a_k \rightarrow v_k\}$ The function mapping a_1 to v_1, a_2 to v_2 etc.

Dom(F) Domain of F.

CoDom(F) Range of F.

$\langle a_1, a_2, \ldots a_k \rangle$ A k-tuple.

$\{e_1, e_2, \ldots e_k\}$ An enumerated set.

$\mathbf{A} \setminus \mathbf{B}$ The set difference of **A** and **B**.

$\{x \mid P(x)\}$ The set of x satisfying P.

$\langle x \mid P(x) \rangle$ A tuple built of all x's satisfying P. The order of the elements is arbitrary unless an order is defined in an accompanying definition.

$e \bowtie \langle e_1 \ldots e_n \rangle$ The tuple formed by prepending e to the given tuple.

$e \bowtie \{\langle e_{1_1} \ldots e_{1_n} \rangle, \langle e_{2_1} \ldots e_{2_n} \rangle \ldots\}$ The set of tuples which can be formed by prepending e to the start of an element of the set, i.e. $\{\langle e, e_{1_1} \ldots e_{1_n} \rangle, \langle e, e_{2_1} \ldots e_{2_n} \rangle \ldots\}$.

Given an entity 'θ' of the form '$\langle A, B \rangle$' we will use 'A_θ' for the first element of 'θ' and 'B_θ' for the second; when 'θ' is fixed by the context we will sometimes leave it off.

§2 Basic Definitions

In this section we will define the most primitive concepts of the representation. These are 'entities', 'terms', 'contexts' and 'views'. These structures are built out of standard set theoretic elements, such as tuples and functions; in addition the structures will use six distinct, unanalysed constants as follows.

TRUE, FALSE

These are just the simple boolean truth values.

UNDEFINED

This is used to indicate that a term has no value in a description.

BOOLEAN, MODAL, DESCRIPTIVE

These are used to distinguish between the three types of term of which state descriptions can be built —

Boolean terms represent simple properties and relations. Such terms are assigned a value of TRUE, FALSE or UNDEFINED by each state of affairs. Examples might be representations of "block37 is blue" or "block37 is inside box7".

Modal terms are assigned as values (descriptions of) states of affairs. The representation of "John believes" would be such a term.

Descriptive terms are used to build up descriptions of actions. Such terms are not assigned values, rather a collection of such terms is used as the representation of a specific action. For instance we might represent the action "John puts block37 in box7" by the collection of 4 descriptive terms "agent(John), action(put), patient(block37), place(box7)".

each term is therefore associated with one of the constants BOOLEAN, MODAL or DESCRIPTIVE.

§2.1 A Universe

A 'universe' is a 4-tuple —

$\langle$ **entities**, **functors**, Arity, Ptype $\rangle$
where **entities** and **functors** are disjoint sets,
Arity : **functors** $\rightarrow \mathbb{N}$
Ptype : **functors** $\rightarrow$ { BOOLEAN, MODAL, DESCRIPTIVE }

The intended meanings of the components of this structure are as follows; members of $\mathbf{entities}_U$ are the primitive concepts in U (for instance 'John', 'John's concept of Bill'), $\mathbf{functors}_U$ is the set of properties and relations in U, Arity_U gives the number of entities involved in a relation and Ptype_U is a partitioning of $\mathbf{functors}_U$ into the three types described in the last section.

For example —

$$\begin{aligned}\text{EXAMPLE} = \langle\, &\{\, \text{A}, \text{B}, \text{C}, \text{FRED}, \text{X}, \text{Y}, \,\},\\ &\{\, \text{BLUE}, \text{ON}, \text{BELIEVES} \,\},\\ &\{\, \text{BLUE} \rightarrow 1,\\ &\quad \text{ON} \rightarrow 2,\\ &\quad \text{BELIEVES} \rightarrow 1 \,\},\\ &\{\, \text{BLUE} \rightarrow \text{BOOLEAN},\\ &\quad \text{ON} \rightarrow \text{BOOLEAN},\\ &\quad \text{BELIEVES} \rightarrow \text{MODAL} \,\} \,\rangle\end{aligned}$$

§2.2 Terms

Given some set of entities from a universe we can build terms from the functors and entities as follows —

For $\mathbf{e} \subseteq \mathbf{entities}_U$ define

$$\begin{aligned}\text{Terms}_U(\mathbf{e}) = \{\, &\langle p, e_1, e_2, \ldots e_k \rangle \,\}\\ &\text{where } p \in \mathbf{functors}_U,\\ &\quad \text{Arity}_U(p) = k,\\ &\quad e_i \in \mathbf{e}\end{aligned}$$

The following are possible terms, members of $\text{Terms}_{\text{EXAMPLE}}(\{\text{A}, \text{B}, \text{C}, \text{FRED}\})$

$$\langle \text{BLUE}, \text{A} \rangle$$

$$\langle \text{ON}, \text{A}, \text{B} \rangle$$

$$\langle \text{BELIEVES}, \text{FRED} \rangle$$

Terms will have types based on the type of the functor; for ease of reference we will define a single function which will give the type of terms or bare functors.

For $p \in \textbf{functors}_U$ define

$$\text{Type}_U(p) = \text{Ptype}_U(p)$$
$$\text{Type}_U(\langle p, e_1, e_2, \ldots e_k \rangle) = \text{Ptype}_U(p)$$

Given the type of a term we can define the possible values which it can be assigned. In a system which is to allow complex world descriptions to be the values of terms it is necessary to specify which entities are available for use in the construction of these descriptions, so the set of possible values for a term must be defined relative to a set of entities[1] —

For $\textbf{e'} \subseteq \textbf{entities}_U$, $t \in \text{Terms}_U(\textbf{e'})$ and $\textbf{e} \subseteq \textbf{entities}_U$ define

$$\begin{aligned}\text{Termvalues}_U(t, e) = {} & \text{if } \text{Type}_U(t) = \text{BOOLEAN} \text{ then } \{ \text{TRUE}, \text{FALSE} \} \\ & \text{if } \text{Type}_U(t) = \text{MODAL} \text{ then } \textbf{entities}_U \cup \text{Views}_U(\textbf{e}) \\ & \text{if } \text{Type}_U(t) = \text{DESCRIPTIVE} \text{ then } \{ \text{TRUE} \}\end{aligned}$$

In order to be able to easily talk about complex modally qualified states of affairs, we now define a 'complex term' as a representation of a chain of modalities and a term which they qualify; for example "John believes that Mary believes that the ball is red". Such complex terms are necessary to represent combinations of facts which the system may need to manipulate as a unit, for instance to detect conflicting goals in a plan.

For $\textbf{e} \subseteq \textbf{entities}_U$ define

---

[1] Views_U is defined in the next section

$$\text{Cterms}_U(\mathbf{e}) = \{\, \langle t_1, t_2, \ldots t_k \rangle \mid \forall i \leq k, t_i \in \text{Terms}_U(\mathbf{e}) \wedge \forall i < k, \text{Type}_U(t_i) = \text{MODAL} \,\}$$

The type of a complex term is the type of its last element.. We shall often refer to the combination of a boolean complex term and its value as a complex fact.

Using this concept, the state of affairs "John believes that Mary believes that the ball is red" would be represented as

$$\langle \langle \text{BELIEVE}, \text{JOHN} \rangle, \langle \text{BELIEVE}, \text{MARY} \rangle, \langle \text{RED}, \text{BALL} \rangle \rangle$$

assuming that the entities and functors involved are present in the universe and have the correct types and arities.

§2.3 Views

A 'view' is a complex structure used to represent a state of affairs. Each view contains a description of some relationships and a mapping which shows how the entities in the model are to be interpreted. See section §4.3 of chapter 5.

To define the structure of a view it is easiest to use several intermediate definitions. First a description of a state of affairs involving a given set of entities.

For $\mathbf{e} \subseteq \mathbf{entities}_U$ define

$$\text{Descriptions}_U(\mathbf{e}) = \{\, \text{D} \mid \text{D} :_{\text{part}} \text{Terms}_U(\mathbf{e}) \rightarrow \{\, \text{TRUE}, \text{FALSE} \,\} \cup \text{Views}_U(\mathbf{e}) \wedge \text{D}(t) \in \text{Termvalues}_U(t, \mathbf{e}) \,\}$$

A possible member of $\text{Descriptions}_{\text{example}}(\{\text{A}, \text{B}, \text{C}, \text{FRED}\})$ is

$$\{\, \langle \text{BLUE}, \text{A} \rangle \rightarrow \text{TRUE}, \langle \text{BELIEVES}, \text{FRED} \rangle \rightarrow \langle \ldots \rangle \,\}$$

Where "$\langle \ldots \rangle$" is a member of $\text{Views}_{\text{example}}(\{\text{A}, \text{B}, \text{C}, \text{FRED}\})$.

From descriptions we can define partial models of the world, called here contexts, which consist of a set of entities and a description of their properties and the relationships between them.

$$\begin{aligned}\mathbf{contexts}_U = \{\, \langle\, \mathbf{e}, \mathrm{D} \,\rangle \mid\ & \\ & \mathbf{e} \subseteq \mathbf{entities}_U \wedge \\ & \mathrm{D} \in \mathrm{Descriptions}_U(\mathbf{e}) \,\}\end{aligned}$$

A member of $\mathbf{contexts}_{\mathrm{example}}$, built from the example description above, is

$$\begin{array}{l}\langle\, \{\, \mathrm{X}, \mathrm{Y}, \mathrm{BILL} \,\}, \\ \quad \{\, \langle\, \mathrm{BLUE}, \mathrm{X} \,\rangle \rightarrow \mathrm{TRUE} \\ \qquad \langle\, \mathrm{BELIEVES}, \mathrm{BILL} \,\rangle \rightarrow \langle \ldots \rangle \,\} \\ \quad \rangle\end{array}$$

Now we can define a 'view'. Given a set of entities, a view is a description of a state of affairs involving those entities. A 'view' differs from a 'description' in that rather than directly ascribing values to terms it is built from a model of the world, represented as a 'context', and a note of how the entities in the model are to be interpreted. This interpretation note is a mapping from the entities in the context to the entities over which the view is defined.

For $\mathbf{e} \subseteq \mathbf{entities}_U$ define

$$\begin{aligned}\mathrm{Views}_U(\mathbf{e}) = \{\, \langle\, c, \mathbf{m} \,\rangle \mid\ & \\ & c \in \mathbf{contexts}_U \wedge \\ & \mathbf{m} \subseteq \mathbf{e} \times \mathbf{e}_c \,\}\end{aligned}$$

Recall that $\mathbf{e}_c$ is element $\mathbf{e}$ of c, that is, the entity set of the context.

For example, here is a member of $\mathrm{Views}_{\mathrm{example}}(\{\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{FRED}\})$ —

$$\begin{array}{l}\langle\, \langle\, \{\, \mathrm{X}, \mathrm{Y}, \mathrm{BILL} \,\}, \\ \quad \{\, \langle\, \mathrm{BLUE}, \mathrm{X} \,\rangle \rightarrow \mathrm{TRUE} \\ \qquad \langle\, \mathrm{BELIEVES}, \mathrm{BILL} \,\rangle \rightarrow \langle \ldots \rangle \,\} \\ \quad \rangle, \\ \quad \{\, \langle\, \mathrm{B}, \mathrm{X} \,\rangle, \\ \quad \langle\, \mathrm{C}, \mathrm{Y} \,\rangle, \\ \quad \langle\, \mathrm{FRED}, \mathrm{BILL} \,\rangle \,\} \\ \quad \rangle\end{array}$$

Another is

$\langle \langle \{ \text{X}, \text{Y}, \text{BILL} \},$
$\{ \langle \text{BLUE}, \text{X} \rangle \rightarrow \text{TRUE}$
$\langle \text{BELIEVES}, \text{BILL} \rangle \rightarrow \langle \ldots \rangle \}$
$\rangle,$
$\{ \langle \text{C}, \text{X} \rangle,$
$\langle \text{B}, \text{Y} \rangle,$
$\langle \text{FRED}, \text{BILL} \rangle \}$
$\rangle$

§3 Manipulating the structures

Below we define the simplest forms of manipulation on the types of structure we have defined.

§3.1 Substitution '/'

Since different contexts describe states of affairs in terms of different sets of entities we need to be able to talk about contexts, descriptions and views which are equivalent apart from the set of entities of which they are built. To do this we define a substitution operation / which we define as follows.

For terms, substitution is done by replacing the entities in the tuple with their counterparts. This also handles the crude 'second order' place holders we will introduce in section §3.2.4 of appendix B. Such place holders are represented as terms of the form $\langle \text{PH}, \text{C} \rangle$; the substitution for C in such a term will always be a term.

For $\mathbf{e}, \mathbf{e}' \subseteq \mathbf{entities}_U$, $\langle p, e_1, \ldots e_i \rangle \in \text{Terms}_U(\mathbf{e})$ and $\phi : \mathbf{e} \rightarrow \mathbf{e}' \cup \text{Terms}_U(\mathbf{e}')$ define

$\langle \text{PH}, e \rangle / \phi = \phi(e)$
$\langle p, e_1, \ldots e_i \rangle / \phi = \langle p, \phi(e_1), \ldots \phi(e_i) \rangle$

For descriptions / maps each term, value pair.

For $\mathbf{e}, \mathbf{e}' \subseteq \mathbf{entities}_U$, $\text{D} \in \text{Descriptions}_U(\mathbf{e})$ and $\phi : \mathbf{e} \rightarrow \mathbf{e}' \cup \text{Terms}_U(\mathbf{e})$ define

$$\mathrm{D}/\phi = \{\, \langle t/\phi, v/\phi \rangle \mid \langle t, v \rangle \in \mathrm{D} \,\}$$

Contexts can be substituted by changing the entity set and the substituting the description.

For $\mathbf{e} \subseteq \mathbf{entities}_U$, $c \in \mathbf{contexts}_U$ and $\phi : \mathbf{e}_c \rightarrow \mathbf{e} \cup \mathrm{Terms}_U(\mathbf{e})$ define

$$c/\phi = \langle\, \{\, \phi(e) \mid e \in \mathbf{e}_c \,\}, \mathrm{D}_c/\phi \,\rangle$$

Substitution on views is done on the mapping.

For $\mathbf{e}, \mathbf{e}' \subseteq \mathbf{entities}_U$, $v \in \mathrm{Views}_U(\mathbf{e})$ and $\phi : \mathbf{e} \rightarrow \mathbf{e}'$ define

$$v/\phi = \langle\, c_v, \{\, \langle e/\phi, e' \rangle \mid \langle e, e' \rangle \in \mathrm{M}_v \,\} \,\rangle$$

For anything else / is an identity operation

$$x/\phi = x$$

If $\phi = \{\textsc{a} \rightarrow \textsc{m}, \textsc{b} \rightarrow \textsc{n}, \textsc{fred} \rightarrow \textsc{mary}\}$ then then the following examples of substitution are valid

A term —

$$\langle\, \textsc{blue}, \textsc{a} \,\rangle/\phi = \langle\, \textsc{blue}, \textsc{m} \,\rangle$$

A Description —

$$\begin{aligned}
&\{\, \langle\, \textsc{blue}, \textsc{x} \,\rangle \rightarrow \textsc{true} \\
&\quad \langle\, \textsc{believes}, \textsc{fred} \,\rangle \rightarrow \langle \ldots \rangle \\
&\quad \}/\phi = \{\, \langle\, \textsc{blue}, \textsc{m} \,\rangle \rightarrow \textsc{true} \\
&\qquad\qquad \langle\, \textsc{believes}, \textsc{mary} \,\rangle \rightarrow \langle \ldots \rangle/\phi \\
&\qquad\qquad \}
\end{aligned}$$

A Context —

$$\begin{aligned}
&\langle\, \{\, \textsc{b}, \textsc{fred} \,\}, \{\, \langle\, \textsc{on}, \textsc{fred}, \textsc{b} \,\rangle \rightarrow \textsc{true} \,\} \\
&\quad \rangle/\phi = \langle\, \{\, \textsc{n}, \textsc{mary} \,\}, \\
&\qquad\qquad \{\, \langle\, \textsc{on}, \textsc{mary}, \textsc{n} \,\rangle \rightarrow \textsc{true} \,\} \\
&\qquad\qquad \rangle
\end{aligned}$$

A view —

$$\begin{aligned}
&\langle\langle\{\text{X},\text{Y}\},\\
&\quad\{\langle\text{BLUE},\text{X}\rangle \rightarrow \text{TRUE}\}\\
&\quad\rangle,\{\langle\text{X},\text{A}\rangle\langle\text{Y},\text{B}\rangle\}\\
&\rangle/\phi = \langle\langle\{\text{X},\text{Y}\},\\
&\qquad\{\langle\text{BLUE},\text{X}\rangle \rightarrow \text{TRUE}\}\\
&\qquad\rangle,\{\langle\text{X},\text{M}\rangle\langle\text{Y},\text{N}\rangle\}\\
&\qquad\rangle
\end{aligned}$$

§3.2 Union of structures: '$\oplus$'

One common manipulation which must be performed in Riple is the merging of a context into the current state of the world (see section §5.2.3.4 of chapter 5). A similar operation must be performed when finding the value of a term in a view, since two terms in the context might have separate values while the mapping of the view makes them both equivalent. In both cases we call the operation 'union' and represent it by the operator $\oplus$. The question arises of what should be done when this operation is applied to views which are incompatible, ie which assign opposing truth values to the same term. This can not arise when it is used to merge states into the current state of the world, since the planning algorithm ensures that such states are always compatible. When merging views in the course of extracting the value of a term, the correct behaviour seems to be to produce a new view which assigns no truth value to the term which was incompatibly defined. To see this consider a case where an agent has incompatible beliefs about the beliefs of two other agent, say the system believes that John believes his car is blue while Jack believes that John's car is not blue. If the system determines that John and Jack are the same entity, it seems clear that the resulting state should contain no information about his beliefs about his car being blue or not.

We can combine two state descriptions to form a new one which carries all the information of both. For contexts the combination can simply be a new context with an entity corresponding to each which exists in either of the component contexts. The description of the new context is built from the descriptions of the components taking into account which of the entities in the new context corresponds to which in the components.

For $c_1, c_2 \in \mathbf{contexts}_U$ define

$$c_1 \oplus c_2 = \langle \mathbf{e}, \mathrm{D} \rangle$$
where
$$\exists \phi, \text{ a bijection from } \mathbf{e}_{c_1} \cup \mathbf{e}_{c_2} \text{ to } \mathbf{e} \text{ s.t. } \mathrm{D} = \mathrm{D}_{c_1}/\phi \cup \mathrm{D}_{c_2}/\phi$$

For example, if we combine the two contexts

$$c_1 = \langle \{ \mathrm{A}, \mathrm{B} \},$$
$$\{ \langle \mathrm{ON}, \mathrm{A}, \mathrm{B} \rangle \rightarrow \mathrm{TRUE}$$
$$\langle \mathrm{BLUE}, \mathrm{B} \rangle \rightarrow \mathrm{FALSE} \} \rangle$$

$$c_2 = \langle \{ \mathrm{FRED}, \mathrm{X} \},$$
$$\{ \langle \mathrm{BELIEVES}, \mathrm{FRED} \rangle \rightarrow \langle \langle \{ \mathrm{Y} \}, \{ \langle \mathrm{BLUE}, \mathrm{Y} \rangle \rightarrow \mathrm{TRUE} \} \rangle$$
$$\{ \langle \mathrm{X}, \mathrm{Y} \rangle \} \rangle$$
$$\} \rangle$$

Then when we follow the above definition we get.

$$c_1 \oplus c_2 = \langle \{ \mathrm{M}, \mathrm{N}, \mathrm{O}, \mathrm{P} \}$$
$$\{ \langle \mathrm{ON}, \mathrm{M}, \mathrm{N} \rangle \rightarrow \mathrm{TRUE}$$
$$\langle \mathrm{BLUE}, \mathrm{N} \rangle \rightarrow \mathrm{TRUE}$$
$$\langle \mathrm{BELIEVES}, \mathrm{O} \rangle \rightarrow$$
$$\langle \langle \{ \mathrm{Y} \}, \{ \langle \mathrm{BLUE}, \mathrm{Y} \rangle \rightarrow \mathrm{TRUE} \} \rangle, \{ \langle \mathrm{P}, \mathrm{Y} \rangle \} \rangle \} \rangle$$

Here the bijection, 'ϕ', is

$$\{ \mathrm{A} \rightarrow \mathrm{M},$$
$$\mathrm{B} \rightarrow \mathrm{N},$$
$$\mathrm{FRED} \rightarrow \mathrm{O},$$
$$\mathrm{X} \rightarrow \mathrm{P} \}$$

In a similar way we can define the combination of two views to form a view expressing the information contained in both. Using the context union operation above this can be expressed simply as —

For $v_1, v_2 \in \mathrm{Views}_U(\mathbf{e})$ define

$$v_1 \oplus v_2 = \langle c_{v_1} \oplus c_{v_2},$$
$$\{ \langle e_1/\phi, e_2/\phi \rangle \mid \langle e_1, e_2 \rangle \in \mathrm{M}_{v_1} \cup \mathrm{M}_{v_2} \} \rangle$$
where ϕ is the bijection used to form $c_{v_1} \oplus c_{v_2}$

For example, if we have the following views from $\text{Views}_{\text{example}}(\{\text{X}, \text{Y}\})$

$$v_1 = \langle \langle \{ \text{A}, \text{B} \}, \\ \{ \langle \text{ON}, \text{A}, \text{B} \rangle \rightarrow \text{TRUE} \} \rangle, \\ \{ \langle \text{X}, \text{A} \rangle \langle \text{Y}, \text{B} \rangle \} \rangle$$

$$v_2 = \langle \langle \{ \text{FRED} \}, \\ \{ \langle \text{BLUE}, \text{FRED} \rangle \rightarrow \text{FALSE} \} \rangle, \\ \{ \langle \text{Y}, \text{FRED} \rangle \} \rangle$$

We can combine them to get

$$v_1 \oplus v_2 = \langle \langle \{ \text{M}, \text{N}, \text{O} \}, \\ \{ \langle \text{ON}, \text{M}, \text{N} \rangle \rightarrow \text{TRUE} \\ \langle \text{BLUE}, \text{O} \rangle \rightarrow \text{FALSE} \} \rangle \\ \{ \langle \text{X}, \text{M} \rangle \langle \text{Y}, \text{N} \rangle \langle \text{Y}, \text{O} \rangle \} \rangle$$

§3.3 Concatenation of Views: '$\odot$'

Another important manipulation on view is combination "in series", so that the state described by the combination is that described by the second view, as viewed through the mapping of the first view. We call this 'concatenation' and represent it by the operator $\odot$. The aim is to take a situation which looks like A–1 where there are two nested views and produce a single view which is equivalent to the combined effect of both. This effect is obtained by combining the mappings of the two views head to tail. This type of combination is used to obtain the value of a modal term in a view 'v', the value will be the value of the term in the context of the view, concatenated with 'v'.

For $\mathbf{e} \subseteq \mathbf{entities}_U$, $v_1 \in \text{Views}_U(\mathbf{e})$ and $v_2 \in \text{Views}_U(e_{c_{v_1}})$ define

$$v_1 \odot v_2 = \langle c_{v_2}, \\ \{ \langle a, c \rangle \mid \\ \exists b \text{ s.t. } \langle a, b \rangle \in \text{M}_{v_1} \wedge \\ \langle b, c \rangle \in \text{M}_{v_2} \} \rangle$$

For example, from

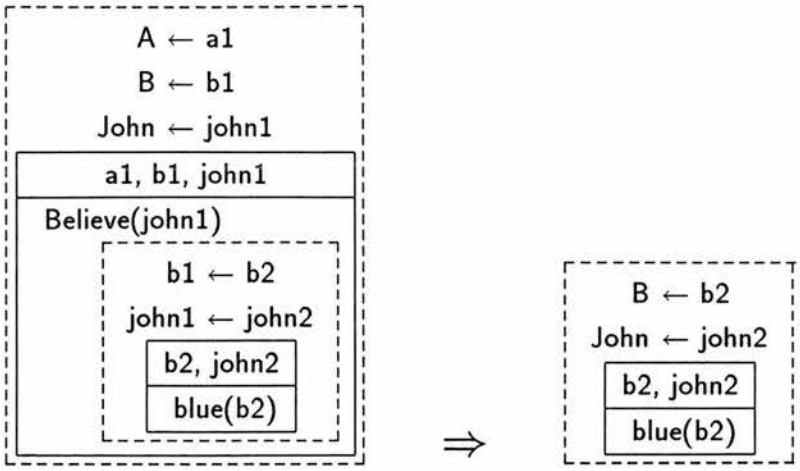


Figure A–1: Two Nested Views and the Result of Concatenating them

$$v_1 = \langle \langle \{ \text{A}, \text{B} \}, \{ \langle \text{ON}, \text{A}, \text{B} \rangle \rightarrow \text{TRUE} \} \rangle, \{ \langle \text{X}, \text{A} \rangle, \langle \text{Y}, \text{B} \rangle \} \rangle$$

and

$$v_2 = \langle \langle \{ \text{FRED} \}, \{ \langle \text{BLUE}, \text{FRED} \rangle \rightarrow \text{FALSE} \} \rangle, \{ \langle \text{B}, \text{FRED} \rangle \} \rangle$$

We get the concatenation

$$v_1 \odot v_2 = \langle \langle \{ \text{FRED} \}, \{ \langle \text{BLUE}, \text{FRED} \rangle \rightarrow \text{FALSE} \} \rangle, \{ \langle \text{Y}, \text{FRED} \rangle \} \rangle$$

§3.4 Taking The View of Something

The final operation which we shall need later in this chapter is that of applying the mapping of a view to an object. This operation will be required each time an object *within* a view must be examined by processes looking 'from outside'.In general for any object there will be more than one possible result of applying a mapping, so we define the function Viewsof which associates with any object and view a set of objects which might be obtained by applying the mapping defined by the view to the object.

For entities this is simple, we just use the mapping which is part of the view

For $\mathbf{e} \subseteq \mathbf{entities}_U$, $v \in \mathrm{Views}_U(\mathbf{e})$ and $e \in \mathbf{e}_{c_v}$ define

$$\mathrm{Viewsof}(e, v) = \{\, e' \mid \langle e', e \rangle \in \mathrm{M}_v \,\}$$

For terms we take all possible ways of viewing the entities over which the term is defined.

For $\mathbf{e} \subseteq \mathbf{entities}_U$, $v \in \mathrm{Views}_U(\mathbf{e})$ and $\langle p, e_1, e_2, \ldots e_k \rangle \in \mathrm{Terms}_U(\mathbf{e})$ define

$$\mathrm{Viewsof}(\langle\, p, e_1, e_2, \ldots e_k \,\rangle, v) = \{\, \langle p, e'_1, e'_2, \ldots e'_k \rangle \mid \forall i, \langle e'_i, e_i \rangle \in \mathrm{M}_v \,\}$$

For views, taking a further view is done by concatenation.

For $\mathbf{e} \subseteq \mathbf{entities}_U$, $v_1 \in \mathrm{Views}_U(\mathbf{e})$ and $v_2 \in \mathrm{Views}_U(\mathbf{e}_{C_{v_1}})$ define

$$\mathrm{Viewsof}(v_2, v_1) = \{\, v_1 \odot v_2 \,\}$$

For any other x define

$$\mathrm{Viewsof}(x, v) = \{\, x \,\}$$

§4 Interpreting the Representation

One basic inquiry which we can make about a representation of this kind is to ask the value of some term in the state of affairs described by some view. The complexities of this problem were glossed over in chapter 5, here we define the process in detail.

Once again we can build up to the final definition in stages.

First we define the value of a term in a context, then how to find all the the terms which may be given values by the context which is part of a view and which are the counterparts of the term we are enquiring about under the transformation defined by the view. Given these two definitions we can define the set of values which a term is assigned by a view, and from this we can determine the aggregate effect of all these assignments. Finally we must take the determined value and apply the transformation defined by the view so as to have the final value for the term expressed as needed by an external questioner, not as expressed 'inside' the view.

§4.1 Inquiring About Terms in a Context

For any member of $\mathbf{contexts}_U$ we represent the value it assigns to a term in the state of affairs it describes by the following function

For $c \in \mathbf{contexts}_U$ and $t \in \mathrm{Terms}_U(\mathbf{e}_c)$ define

$$\mathrm{Assigned\_Val}(c,t) = \text{if } t \in \mathrm{Dom}(\mathrm{D}_c) \text{ then } \mathrm{D}_c(t) \text{ otherwise } \text{UNDEFINED}$$

That is, if the term is in the domain of the partial function which forms the 'description' of the context we use that to get the definition, otherwise the term has no value.

§4.2 Inquiring About Terms in a View

The definitions in this section provide the means of determining the value which a view assigns to a term.

§4.2.1 Finding Counterparts

Since the context part of a view describes the world using its own, private set of entities, the first thing which must be done when trying to determine the value of a term in a view is to decide which terms in the model of the world represented by the view's context would correspond to the term we are inquiring about. This is defined by the following function which is similar to 'Viewsof' except that it uses the view's mapping in reverse.

For $\mathbf{e} \subseteq \mathbf{entities}_U$, $v \in \mathrm{Views}_U(\mathbf{e})$ and $\langle p, e_1, e_2, \ldots e_k \rangle \in \mathrm{Terms}_U(\mathbf{e}_{c_v})$ define

$$\mathrm{Rev\_Viewsof}(v, \langle p, e_1, e_2 \ldots e_k \rangle) = \{ \langle p, e'_1, e'_2 \ldots e'_k \rangle \mid \forall i, \langle e_i, e'_i \rangle \in \mathrm{M}_v \}$$

So, for example,if

$$v = \langle \langle \{ \text{A}, \text{B}, \text{C} \}, \\ \{ \langle \text{ON}, \text{A}, \text{B} \rangle \rightarrow \text{TRUE}, \\ \langle \text{ON}, \text{B}, \text{C} \rangle \rightarrow \text{FALSE} \} \rangle, \\ \{ \langle \text{X}, \text{A} \rangle \langle \text{X}, \text{B} \rangle \langle \text{Y}, \text{B} \rangle \langle \text{Y}, \text{C} \rangle \} \rangle$$

then

$$\text{Rev\_Viewsof}(v, \langle \text{ON}, \text{X}, \text{Y} \rangle) = \\ \{ \langle \text{ON}, \text{A}, \text{B} \rangle \langle \text{ON}, \text{A}, \text{C} \rangle \langle \text{ON}, \text{B}, \text{B} \rangle \langle \text{ON}, \text{B}, \text{C} \rangle \}$$

Clearly $\langle \text{ON}, \text{B}, \text{B} \rangle$ is not a sensible term, under the obvious interpretation. It would be possible to filter out certain classes of term in the manipulations, but if we are to avoid creating an active representation, as was decided in section §3 of chapter 3, it is not possible to include a more general system of integrity constraints.

§4.2.2 Possible Values in a View

We can the find out a term's possible values by looking at the values which the view's context assigns to each of its counterparts .

For $\mathbf{e} \subseteq \mathbf{entities}_U$, $v \in \text{Views}_U(\mathbf{e})$ and $term \in \text{Terms}_U(\mathbf{e}_{c_v})$ define

$$\text{Values}(v, term) = \{ val \mid \exists t \in \text{Rev\_Viewsof}(v, term) \text{ s.t.} \\ val = \text{Assigned\_Val}(c_v, t) \wedge \\ val \neq \text{UNDEFINED} \}$$

Thus, taking the above example again,

$$\text{Values}(v, \langle \text{ON}, \text{X}, \text{Y} \rangle) = \\ \{ \text{TRUE}, \text{FALSE} \}$$

Since the terms '$\langle \text{ON}, \text{A}, \text{C} \rangle$' and '$\langle \text{ON}, \text{B}, \text{B} \rangle$' have the value 'UNDEFINED' while '$\langle \text{ON}, \text{A}, \text{B} \rangle$' and '$\langle \text{ON}, \text{B}, \text{C} \rangle$' have values 'TRUE' and 'FALSE' respectively the context of view 'v'.

§4.2.3 Combining Values

Once we have a collection of values for a term we can determine what the aggregate effect of these definitions is. The following function describes the method of combination of a number of values into one.

For $\mathbf{e} \subseteq \mathbf{entities}_U$, $v \in \text{Views}_U(\mathbf{e})$ and $term \in \text{Terms}_U(\mathbf{e}_{c_v})$ define

$$\begin{aligned}
&\text{Value_In_View}(v, term) = \\
&\qquad \text{if } \text{Type}_U(term) = \text{BOOLEAN} \text{ then} \\
&\qquad\quad \text{if } \text{Values}(v, term) = \{\, val \,\} \text{ then} \\
&\qquad\qquad val \\
&\qquad\quad \text{otherwise} \\
&\qquad\qquad \text{UNDEFINED} \\
&\qquad \text{otherwise if } \text{Type}_U(term) = \text{MODAL} \text{ then} \\
&\qquad\quad v_1 \oplus v_2 \oplus \ldots \oplus v_k \\
&\qquad\quad \text{where} \\
&\qquad\qquad \text{Values}(v, term) = \{\, v_1, v_2, \ldots v_k \,\}
\end{aligned}$$

This states that boolean terms are undefined when they have counterpart terms in the view with more than one value. Modal terms, on the other hand, have a value which is the union of all the values of their counterparts. To continue the above example,

$$\text{Value\_In\_View}(v, \langle \text{ON}, \text{X}, \text{Y} \rangle) = \text{UNDEFINED}$$

Since '$\langle$ON, X, Y$\rangle$' is boolean and has counterparts with differing values.

§4.2.4 From Value to Definition

The actual definition of a term in a view is the result of passing the above value through the mapping function of the view as follows.

For $\mathbf{e} \subseteq \mathbf{entities}_U$, $v \in \text{Views}_U(\mathbf{e})$ and $term \in \text{Terms}_U(\mathbf{e}_{c_v})$ define

$$\begin{aligned}
&\text{Assigned_Val}(v, term) = \text{if } \text{Type}_U(term) = \text{BOOLEAN} \text{ then} \\
&\qquad\qquad \text{Value_In_View}(v, term) \\
&\qquad \text{otherwise if } \text{Type}_U(term) = \text{MODAL} \text{ then} \\
&\qquad\quad v_1 \oplus v_2 \oplus \ldots \oplus v_k \\
&\qquad\quad \text{where} \\
&\qquad\qquad \text{Viewsof}(\text{Value_In_View}(v, term), v) \\
&\qquad\qquad\quad = \{\, v_1, v_2 \ldots v_k \,\}^2
\end{aligned}$$

---

[2]In fact the definition of 'Viewsof' ensures that this will always be a singleton set

§4.3 Inquiring About Complex Terms in Contexts and Views

Complex terms (that is sequences of terms representing nested modalities) are also be assigned values by a view. We will need to determine this value when looking for conflicts in a plan or for states in the plan which 'match' the left and side of a rule.

The definition of the value of a complex term in a view is given by the recursive definitions of this section. It should be noted that in the cases where one of the modal terms in the complex term is assigned an entity as a value by the view being examined, the value of the complex term in that view is undefined. This causes no problems since entities as modal term values are only used in the system in the representation of actions and rules, they are never inserted into the plan and so are never interpreted using these definitions. See section §3.2.4 of appendix B for a description of how this internal representation is used.

§4.3.1 Definitions for Simple 'Complex' Terms

First the base case, a 'complex' term which consists of a single term is defined in just the same way as that fact would be;

For $\mathbf{e} \subseteq \mathbf{entities}_U$, $x \in \text{Views}_U(\mathbf{e}) \cup \mathbf{contexts}_U$ and $t \in \text{Terms}_U(\mathbf{e})$ define

$$\text{Assigned\_Val}(x, \langle\, t \,\rangle) = \text{Assigned\_Val}(x, t)$$

§4.3.2 The Definition of a Complex Term in a Context

The definition of a complex term in a context is decided by finding the definition of the first term in the complex term and finding its value. This gives a view which is the context for the evaluation of the rest of the complex term.

For $\mathbf{e} \subseteq \mathbf{entities}_U$, $c \in \mathbf{contexts}_U$ and $\langle t_1, t_2 \ldots t_n \rangle \in \text{Cterms}_U(\mathbf{e})$ define

$$\text{Assigned\_Val}(c, \langle\, t_1, t_2 \ldots t_n \,\rangle) =$$
$$\text{if } v' \notin \mathbf{entities}_U \text{ then}$$
$$\text{Assigned\_Val}(v', \langle\, t_2 \ldots t_n \,\rangle)$$
$$\text{otherwise UNDEFINED}$$

where

$$v' = \text{Assigned\_Val}(c, t_1)$$

For example, if

$$\begin{aligned} c = \langle\, &\{\, \text{FRED}, \text{X} \,\}, \\ &\{\, \langle\, \text{BELIEVES}, \text{FRED} \,\rangle \rightarrow \langle\, \langle\, \{\, \text{A} \,\}, \{\, \langle\, \text{BLUE}, \text{A} \,\rangle \rightarrow \text{FALSE} \,\}\, \rangle \\ &\quad \{\, \langle\, \text{X}, \text{A} \,\rangle \,\}\, \rangle \,\}\, \rangle \end{aligned}$$

then by the above definition,

$$\begin{aligned} &\text{Assigned\_Val}(c, \langle\, \langle\, \text{BELIEVES}, \text{FRED} \,\rangle, \langle\, \text{BLUE}, \text{X} \,\rangle\, \rangle) = \\ &\qquad \text{Assigned\_Val}(\langle\, \langle\, \{\, \text{A} \,\}, \{\, \langle\, \text{BLUE}, \text{A} \,\rangle \rightarrow \text{FALSE} \,\}\, \rangle \\ &\qquad\qquad \{\, \langle\, \text{X}, \text{A} \,\rangle \,\}\, \rangle, \langle\, \langle\, \text{BLUE}, \text{X} \,\rangle\, \rangle) \end{aligned}$$

Since the complex term in the latter expression is a single fact, the base case, defined in section §4.3.1 holds and so

$$\begin{aligned} &\text{Assigned\_Val}(c, \langle\, \langle\, \text{BELIEVES}, \text{FRED} \,\rangle, \langle\, \text{BLUE}, \text{X} \,\rangle\, \rangle) = \\ &\qquad \text{Assigned\_Val}(\langle\, \langle\, \{\, \text{A} \,\}, \{\, \langle\, \text{BLUE}, \text{A} \,\rangle \rightarrow \text{FALSE} \,\}\, \rangle \\ &\qquad\qquad \{\, \langle\, \text{X}, \text{A} \,\rangle \,\}\, \rangle, \langle\, \text{BLUE}, \text{X} \,\rangle) \end{aligned}$$

By using the definitions from section §4.2 one can easily see that the final expression here has value 'FALSE', and so

$$\text{Assigned\_Val}(c, \langle\, \langle\, \text{BELIEVES}, \text{FRED} \,\rangle, \langle\, \text{BLUE}, \text{X} \,\rangle\, \rangle) = \text{FALSE}$$

§4.3.3 The Definition of a Complex Term in a View

The definition of a complex term in a view is built up in the same way as the definition of a term in a view, so these definitions closely parallel those of sections §4.2.2–§4.2.4. First we need to be able to find all the values of a complex term.

For $\mathbf{e} \subseteq \mathbf{entities}_U$, $v \in \text{Views}_U(\mathbf{e})$ and $\langle t_1, t_2 \ldots t_n \rangle \in \text{Cterms}_U(\mathbf{e})$ define

$$\begin{aligned} &\text{Values}(v, \langle\, t_1, t_2 \ldots t_n \,\rangle) = \\ &\qquad \{\, val \mid val = \text{Assigned\_Val}(c_v, \langle\, t'_1, t'_2 \ldots t'_n \,\rangle) \wedge \\ &\qquad\quad val \neq \text{UNDEFINED} \\ &\qquad\quad \text{where} \\ &\qquad\qquad \forall i, t'_i \in \text{Rev\_Viewsof}(v, t_i) \,\} \end{aligned}$$

Next we define how these many values are reduced to one aggregate value.

For $\mathbf{e} \subseteq \mathbf{entities}_U$, $v \in \text{Views}_U(\mathbf{e})$ and $\langle t_1, t_2 \ldots t_n \rangle \in \text{Cterms}_U(\mathbf{e})$ define

$$
\begin{aligned}
&\text{Value\_In\_View}(v, \langle\, t_1, t_2 \ldots t_n \,\rangle) = \\
&\qquad \text{if } \text{Type}_U(t_n) = \text{BOOLEAN} \text{ then} \\
&\qquad\quad \text{if } \text{Values}(v, \langle\, t_1, t_2 \ldots t_n \,\rangle) = \{\, val \,\} \text{ then} \\
&\qquad\qquad val \\
&\qquad\quad \text{otherwise} \\
&\qquad\qquad \text{UNDEFINED} \\
&\qquad \text{otherwise if } \text{Type}_U(t_n) = \text{MODAL} \text{ then} \\
&\qquad\quad v_1 \oplus v_2 \oplus \ldots \oplus v_k \\
&\qquad\quad \text{where} \\
&\qquad\qquad \text{Values}(v, \langle\, t_1, t_2 \ldots t_n \,\rangle) = \{\, v_1, v_2 \ldots v_k \,\}
\end{aligned}
$$

Finally the value must be examined in the light of the view.

For $\mathbf{e} \subseteq \mathbf{entities}_U$, $v \in \text{Views}_U(\mathbf{e})$ and $\langle t_1, t_2 \ldots t_n \rangle \in \text{Cterms}_U(\mathbf{e})$ define

$$
\begin{aligned}
&\text{Assigned\_Val}(v, \langle\, t_1, t_2 \ldots t_n \,\rangle) = \\
&\qquad \text{if } \text{Type}_U(t_n) = \text{BOOLEAN} \text{ then} \\
&\qquad\quad \text{Value\_In\_View}(v, \langle\, t_1, t_2 \ldots t_n \,\rangle) \\
&\qquad \text{otherwise if } \text{Type}_U(t_n) = \text{MODAL} \text{ then} \\
&\qquad\quad v_1 \oplus v_2 \oplus \ldots \oplus v_k \\
&\qquad\quad \text{where} \\
&\qquad\qquad \text{Viewsof}(\text{Value\_In\_View}(v, \langle\, t_1, t_2 \ldots t_n \,\rangle), v) = \{\, v_1, v_2 \ldots v,
\end{aligned}
$$

§4.4 Finding All The 'Effects' of a View

Another important notion is the set of complex terms which are given a value by a view. An informal description of the method used to calculate this set was given in section §5.3.2 of chapter 5. In brief, we recursively search the view breaking cycles by remembering which views we have seen.

For $\mathbf{e} \subset \mathbf{entities}_U$, $v \in \text{Views}_U(\mathbf{e})$ and $\mathbf{old} \subset \text{Views}_U(\mathbf{e})$ define

$$
\begin{aligned}
\text{Cterms\_In}(v, \mathbf{old}) = \bigcup_{t \in \text{Termsin}(v)} \quad & \text{if } \text{Type}_U(t) = \text{BOOLEAN} \text{ then } \{\langle\, t \,\rangle\} \\
& \text{otherwise if } \text{Assigned\_Val}(v, t) = v' \wedge
\end{aligned}
$$

$v' \notin$ **old** then
$t \bowtie \text{Cterms\_In}(v', \textbf{old} \cup \{v'\})$
otherwise $\emptyset$

§5 The Representation Used by Riple

The implemented system has only one major difference from the model presented here. Because of the limited needs of the later parts of the system it was decided to limit the entity to entity mapping which form part of views to be functions, each entity in the context of the view being associated with only one 'outside' entity, rather then general relations as described here. This means that the representations manipulated by the program form a subset of the class of representations defined here.

Appendix B

Planning: Domains, Actions, Rules and Agents

§1 Introduction

The representation defined in appendix A is very general. It was designed to make as few assumptions about the representation of relationships and cross world identity as possible. In chapter 5 we developed a set of constructs for simple planned action in a multi-agent domain guided by the needs of discourse planning. We are now in a position to define these constructs more formally using the tools developed in appendix A.

In addition to defining these constructs we shall describe in this chapter the language which is used in chapter 7 to present models of discourse planning. This language has been designed to have a well defined semantics in terms of the full representation and yet to have a clear, intuitive, meaning.

We define the language, and hence the class of structures which interests us, in stages; starting with the basic elements of the representations — 'constants', 'place holders' and 'functors' — and then moving on to show how these are built to form descriptions of 'actions' and goal formation 'rules'. When these concepts have been introduced we shall describe the notion of an 'agent' and how this is related to beliefs in these structures.

Each section provides both a description of the syntax used to describe a particular object and how this syntax is related to the representational structures defined in

appendix A. The syntax is defined using a form of context free grammar using the following conventions.

A grammar rule has the form:

⟨non terminal⟩ ⇒ *expansion*

where '*expansion*' is a sequence of items built up from the following basic constructs.

`'thing'`	`thing` appears literally in the construct, i.e. it is a terminal of the grammar.
⟨thing⟩	⟨thing⟩ is a non terminal.
A^*	Zero or more occurrences of A
A^+	One or more occurrences of A
A^n	n instances of A (where n is an integer)
A | **B**...	Either A or B or...
{ **X** }	X is optional
(*expansion*)	for grouping

For example the following are valid production rules.

⟨list⟩ ⇒ '[' (⟨atom⟩ | ⟨list⟩)* { '|' ⟨atom⟩ } ']'

⟨atom⟩ ⇒ 'a' | 'b'

Describing a simple notation for nested lists. These rules would allow the following texts to be members of the category '⟨list⟩'

[ ]
[b a | b]
[[a] a [b [b | a]]]

In the descriptions of the meanings of the constructs we will refer to the meanings of various elements in the expansion by using the category names in denotation brackets ' ⟦ ⟧ '. Where there is more than one element of a particular category in an expansion we will use superscripted numbers indicating the position in the expansion (counting from the left) to disambiguate. Thus, given the expansion

$$\langle \mathsf{pair} \rangle \Rightarrow \text{'('} \langle \mathsf{element} \rangle \text{'.'} \langle \mathsf{element} \rangle \text{')'}$$

We could describe the meaning of such a ⟨pair⟩ as being a tuple by saying that

$$[\![\mathsf{pair}]\!] = \langle\, [\![\mathsf{element}]\!]^1, [\![\mathsf{element}]\!]^2 \,\rangle$$

Similarly the actual structure of the element will be referred to using the category name inside angle brackets '⟨ ⟩', disambiguated with numeric superscripts.

We assume the following categories are well defined

⟨integer⟩ The category of all the decimal representations of integers, their meanings are just the integers themselves.

⟨name⟩ The category of names. These are just strings of characters taken from the upper and lower case letters, the digits and the underscore '_'. They are their own meanings, that is

$$[\![\mathsf{name}]\!] = \langle \mathsf{name} \rangle$$

§2 Basic Concepts

This section describes the most basic objects in the model — 'entities', and 'functors', and how they are declared in the description of a planning domain.

§2.1 Planning Domains

Formally a planning domain is a 11-tuple —

$$\langle\, \text{NAME}, U, \textit{basecontext}, \text{Den}, \text{Is\_pred}, \mathbf{constants}, \mathbf{agents}, \mathbf{scenarios}, \mathbf{actions}, \mathbf{rules} \,\rangle$$

where

NAME is a member of the category ⟨**names**⟩.

U is a universe as defined in appendix A.

basecontext is an element of $\mathbf{contexts}_U$.

Den is a one to one (partial-) function mapping some members of the category ⟨**names**⟩ to members of $\mathbf{functors}_U \cup \mathbf{entities}_U$.

Is_pred is a one to one function mapping members of $\mathbf{e}_{basecontext}$ to members of $\mathbf{functors}_U$.

constants, agents, scenarios, actions and **rules** are sets.

The intended meaning of such a structure is as follows: 'NAME' is just an arbitrary label for the domain; 'U' is the universe of entities and properties which is described by the domain; '*basecontext*' is a description of the minimal assumptions which the domain makes about the world, in particular has one entity for each constant or agent of the domain; ' Den' defines the mapping between names and their meanings in descriptions of the domain; ' Is_pred' defines the relationship between a constant and the property of being that constant; '**constants**', '**agents**', '**scenarios**', '**actions**' and '**rules**' are the sets of constructs which the domain makes available for planning.

A planning domain is described by a text of the following form

⟨planning domain⟩ ⇒ 'riple_domain' { ⟨integer⟩ } ⟨name⟩
⟨declaration⟩*
⟨definition⟩*

⟨declaration⟩ ⇒ ⟨constant declaration⟩ |
⟨functor declaration⟩ |
⟨agent declaration⟩

⟨definition⟩ ⇒ ⟨scenario definition⟩ |
⟨action definition⟩ |
⟨rule definition⟩

Such a text describes a planning domain 'D' where NAME_D is the given name; U_D, $basecontext_D$, Den_D, $\textbf{constants}_D$ and $\textbf{agents}_D$ are described by the declarations and $\textbf{scenarios}_D$, $\textbf{actions}_D$ and $\textbf{rules}_D$ are given by the definitions.

The different kinds of declaration are described in §2.2.2 together with the constraints they place on the planning system. The definitions are described in §4.

§2.2 Place Holders, Constants, and Functors

The basic building blocks for forming a description of the world which are made available by the representation presented in the previous chapter are 'functors' and 'entities'. However, in a planning system it is useful to distinguish between entities fulfilling three different roles.

Constants Constants represent definite 'things' in the real world. A constant always represents the same notional 'thing' no matter where is appears. A blocks world planning system might have constants representing a table and various blocks; a language planning system might have constants representing more abstract 'things' such as tenses, as well as other agents and the objects under discussion.

Place Holders Place Holders, on the other hand, represent unspecified 'things'. When a description is used in different contexts the place holders it contains may represent different 'things'. Thus these place holders behave like existentially quantified variables in predicate logic. Also, as described in section §5.3.3 of chapter 5 place holders can sometimes behave as universally qualified variables. As was mentioned in chapter 5 we use the term 'place holder' to refer to entities which are fulfilling this role in the structures built by the system. We avoid the term 'variable' as it has connotations from its use in describing logics and computer languages which are not relevant here.

Agents Agents are like constants except that they represent 'things' in the world which are capable of performing 'actions' and so

changing the state of the world. Agents also have the property that they possess a knowledge of (some subset of) the state of the world. Every planning domain will contain at least one agent, representing the planner itself. Additional agents could be included in a domain to represent other planning system and/or human beings which the planner will have to interact with.

To distinguish these three types of entity we use the idea that for every definite 'thing' there is a property of being that thing. Thus every constant or agent which exists in a particular planning domain will have associated with it a functor of arity one which represents this property. In this way the entities which stand for a particular constant object in the beliefs of different agents are identifiable because those beliefs will contain an explicit statement of the identity of each constant.

The distinction between agents and other constants is more subtle. An entity representing an agent will be manipulated in the same way as one representing a constant. The distinction lies outside the representation in the constraints on the planning system. The structures which can be built by the system are constrained by rules, described in section §5, which restrict the form of views which may be assigned as values to terms of the form `believe(a)` where `a` is an agent. Also the planning algorithm is allowed to plan actions to be performed by entities which are agents.

In the following subsections we will define the syntax used to refer to place holders, constants, and functors and then describe the declarations which are used to describe which members of each class exist in a given domain and what names are to be used for them. Agents are described in section §5.

§2.2.1 The Syntax for Place Holders, Constants and Functors

The syntax which we use for these is as follows —

⟨place holder⟩ ⇒ '?' ⟨name⟩

⟨constant⟩ ⇒ ⟨name⟩

⟨agent⟩ ⇒ ⟨name⟩

⟨functor⟩ ⇒ ⟨name⟩

The meaning of a ⟨place holder⟩, ⟨constant⟩ or an ⟨agent⟩ will be a member of $\mathbf{entities}_U$ for the universe 'U' of the planning domain currently being described; however which entity it represents will depend on the context in which it is used and will be defined later .

The meaning of a ⟨functor⟩ is a member of $\mathbf{functors}_U$ and is given by the function 'Den' in the planning domain applied to the ⟨name⟩.

§2.2.2 Constant Declarations

Since there is a functor associated with each named constant, the system must be told about them. This is achieved by using a 'constants' declaration in the description of a planning domain. The syntax of such a declaration is as follows:

⟨constant declaration⟩ ⇒ 'constants' ⟨name⟩* ';'

This has the effect of constraining the planning domain 'D' being described in the following ways

For each ⟨name⟩ n in the declaration define

$\exists\, e \in \mathbf{e}_{\text{basecontext}_D}$ s.t.
$e \in \mathbf{constants}_D \wedge$
$\text{Den}(n) = e \wedge$
$\text{Is\_Pred}_D(e) = f$
where
$f \in \mathbf{functors}_{U_D} \wedge$
$\text{Arity}_{U_D}(f) = 1 \wedge$
$\text{Ptype}_{U_D}(f) = \text{BOOLEAN}$

For example the declaration

```
constants
    mary
    wendy
    ;
```

would create two entities, say 'M_1' and 'W_1'. in '$\mathbf{e}_{\text{basecontext}_D}$' and would add these to the set '$\mathbf{constants}_D$'. Also it would add two boolean functors of arity 1 to '$\mathbf{functors}_{U_D}$', say 'WENDY_P' and 'MARY_P', and set

$\text{Den}(\text{`mary'}) = \text{M}_1$
$\text{Den}(\text{`wendy'}) = \text{W}_1$
$\text{Is\_Pred}_D(\text{M}_1) = \text{MARY\_P}$
$\text{Is\_Pred}_D(\text{W}_1) = \text{WENDY\_P}$

§2.2.3 Functor Declarations

Functors must be declared so that the system knows the number of them which are to be used, their names, types and arities. Since there are three types of functor (boolean, modal and descriptive) there must be three types of declaration, these are as follows.

⟨boolean declaration⟩ ⇒ 'boolean' (⟨name⟩ '/' ⟨integer⟩)*

⟨modal declaration⟩ ⇒ 'modal' (⟨name⟩ '/' ⟨integer⟩)*

⟨descriptive declaration⟩ ⇒ 'descriptive' (⟨name⟩ '/' ⟨integer⟩)*

The presence of a declaration of type TYPE in the description of a planning domain has the affect of putting the following constraints on the described domain

For each ⟨name⟩ and ⟨integer⟩ pair n/i in the declaration define

$\exists f \in \textbf{functors}_{U_D}$ s.t. $\text{Arity}_{U_D}(f) = i \wedge$
$\text{Ptype}_{U_D}(f) = \text{TYPE} \wedge$
$\text{Den}(n) = f$

The full functor declaration looks like the following:

⟨functor declaration⟩ ⇒ 'functors'
(⟨boolean declaration⟩ |
⟨modal declaration⟩ |
⟨descriptive declaration⟩)*
';'

An example of such a definition might be

```
functors
    boolean
        taller/2
        red/1
    modal
        believes/1
    ;
```

Which would have the effect of creating three more functors in '$\mathbf{functors}_{U_D}$', 'TALLER' with type 'BOOLEAN and arity 2, 'RED' with type 'BOOLEAN' and arity 1 and 'BELIEVES' with type 'MODAL' and arity 1. It would then set

Den('taller') = TALLER
Den('red') = RED
Den('believes') = BELIEVES

§3 Terms and Outlines

A term is built up from a functor and a number of constants, place holders or agents equal to its arity. Such terms are used as the building blocks of descriptions of states of affairs and descriptions of actions, which we call 'outlines'. The syntax and semantics of these two types of construct are described in the next two subsections.

§3.1 Terms

The syntax of ⟨term⟩s is

⟨term⟩ ⇒ ⟨boolean term⟩ |
⟨modal term⟩ |
⟨descriptive term⟩

⟨boolean term⟩ ⇒ ⟨functor⟩ '(' { ⟨arg⟩ (',' ⟨arg⟩)$^{n-1}$ } ')'

⟨modal term⟩ ⇒ ⟨functor⟩ '(' { ⟨arg⟩ (',' ⟨arg⟩)$^{n-1}$ } ')'

⟨descriptive term⟩ ⇒ ⟨functor⟩ '(' { ⟨arg⟩ (',' ⟨arg⟩)$^{n-1}$ } ')'

$$\langle \text{arg} \rangle \Rightarrow \langle \text{place holder} \rangle \mid \langle \text{constant} \rangle \mid \langle \text{agent} \rangle \mid \text{'ME'}$$

Where

$$\begin{aligned} &\text{Arity}_{U_D}([\![\text{functor}]\!]) = n \\ &\text{Ptype}_{U_D}([\![\text{functor}]\!]) = \text{BOOLEAN for } \langle \text{boolean term} \rangle \text{s}, \\ &\qquad\qquad\qquad\qquad \text{MODAL for } \langle \text{modal term} \rangle \text{s}, \\ &\qquad\qquad\qquad\qquad \text{DESCRIPTIVE for } \langle \text{descriptive term} \rangle \text{s} \end{aligned}$$

The meaning of such a construction is the n-tuple

$$\langle\, [\![\text{functor}]\!], [\![\text{arg}]\!]^1, [\![\text{arg}]\!]^2, \ldots [\![\text{arg}]\!]^n \,\rangle$$

For example the term

```
taller(wendy, mary)
```

would have as its meaning the triple

$$\langle\, \text{TALLER}, \text{W}_7, \text{M}_7 \,\rangle$$

if 'W_7' is the meaning ascribed to '`wendy`' by the context and 'M_7' is the meaning ascribed to '`mary`'.

The special term argument '`ME`' is used to refer to the agent whose beliefs are being defined. It is discussed in detail in section §5 but for the present discussion it can be assumed to be just another ⟨constant⟩.

§3.2 Outlines

'Outlines' are the textual representation of views, as described in the previous chapter. As such they are partial descriptions of world states or of actions. Since they do represent views, they consist of two parts, a description of a state of affairs in the form of a context and an interpretation of how the entities in that description are to be interpreted. For the sake of clarity we will describe the way in which the syntax of ⟨outline⟩s corresponds to the views which form their meanings in two stages, first the mapping onto contexts and then the way in which the syntax determines the interpretation of the entities.

Some changes to the meanings described in this section are described in the discussion of representation of beliefs in section §5.2 below.

§3.2.1 The Syntax of Outlines

The syntax of a ⟨outline⟩ is as follows:

⟨outline⟩ ⇒ ⟨action outline⟩ |
⟨state outline⟩

⟨action outline⟩ ⇒ ⟨descriptive term⟩*

⟨state outline⟩ ⇒ { '[' ⟨place holder⟩+ ']' } (
({ '~' } ⟨boolean term⟩) |
⟨modal term⟩ '=' { ⟨place holder⟩ } '{' ⟨state outline⟩ '}' |
⟨modal term⟩ '=' ⟨place holder⟩ |
⟨place holder⟩
)*

§3.2.2 Outlines as Contexts

A ⟨outline⟩ of the form defined above represents a view, 'v'. The context part of this view 'c_v' is defined by the sequence of terms given in the ⟨outline⟩. The mapping of the view is defined by the context in which the ⟨outline⟩ appears and is defined in section §3.2.3. The context itself consists of two parts, a set of entities and a (partial) function giving truth values to some of the facts which can be defined over those entities[1].

The set of entities contains one and only one entity for each constant, place holder or agent used in any term in the syntax of the ⟨outline⟩; this includes nested terms. Thus the ⟨state outline⟩

```
taller ( mary, ?someone )
believes ( mary ) = {
   ~taller ( mary, wendy )
    }
```

would represent a view whose context contains 3 entities, one each for '`mary`', '`?someone`' and '`wendy`'. The nested ⟨state outline⟩ between the braces, on the

---

[1]An added complication is caused by the possibility of having place holders in the outline as shown by the fourth option in the definition of a state outline above. See section §3.2.4

other hand, would represent a view whose context contains only two entities, for 'mary' and 'wendy'.

The (optional) list of ⟨place holders⟩ at the start of a ⟨state description⟩ represents a set of entities which are not to have equivalents in the views which refer to this one, as was discussed in section §5.3.3 of chapter 5. For now we shall just give a way to refer to them in later definitions —

For d an ⟨outline⟩ define

$$\text{Quantified\_In}(d) = \{\, e \mid e = [\![\text{ph}]\!] \wedge \text{ph is a member of the list at the top of } d \,\}$$

The association of entities with ⟨name⟩s is obviously arbitrary. As the outline is read an entity is created for each ⟨name⟩ used. We represent whatever association is produced by the following function.

For d an ⟨outline⟩ s.t. $[\![\text{d}]\!] = v$, n a ⟨name⟩ and $e \in \mathbf{e}_{c_v} \setminus \text{Quantified\_In}(d)$ define

$$\text{Entity\_For}(n, d) = e \text{ where } e \text{ is the entity in the context which was associated with } n \text{ in } d$$

This function gives the meaning to the arguments in the ⟨term⟩s which form the body of the outline.

The Truthvalue function of the context maps those terms which are directly part of the ⟨outline⟩ onto truth values as follows

⟨boolean term⟩ If preceded by '~' these are mapped to FALSE otherwise to TRUE.

⟨modal term⟩ If the '=' is followed (possible after a ⟨place holder⟩) is followed by a ⟨state outline⟩ within braces then the modal term is mapped to the view which is denoted by that state outline. The ⟨place holder⟩ if present is just a label for the ⟨state description⟩.

If the '=' is followed by just a ⟨place holder⟩ then the term is mapped to the view denoted by the ⟨state description⟩

labelled with that ⟨**place holder**⟩. This allows us to describe structures built from views which are not trees.

⟨**descriptive term**⟩ These are all mapped to TRUE since they are simply used as labels for actions.

Additionally, if this ⟨**outline**⟩ represents a belief view (see section §5.4.1 of chapter 5 and section §5 below) then for each constant used in the ⟨**outline**⟩ the truth value function must assign the value TRUE to the assertion that the associated entity has the property of being the named constant. That is —

$$\langle\, \langle\, \text{Is\_Pred}_D(\text{Den}(\text{`}\texttt{name}\text{'})), \text{Entity\_For}(\langle \texttt{name} \rangle)\, \rangle, \text{TRUE}\, \rangle$$

One final thing must be explained. Since the syntax of ⟨**state outline**⟩s allows place holders in place of textual representations of the content of a view, we need to define the scope of these names. Any two uses of the same ⟨**place holder**⟩ within one ⟨**definition**⟩[2] will refer to the same ⟨**state description**⟩. If there is no instance of the ⟨**place holder**⟩ being associated with a real ⟨**state outline**⟩, then the value associated with the modal term will be an entity. See section §3.2.4 for a definition of what this means.

So, given the declarations in the earlier examples, and assuming it is in a context where it is known to be a belief view, the above example outline would represent a view of the following form[3]

$$\begin{array}{l}
\langle\, \langle\, \{\, \text{M}, \text{S}, \text{W}\, \}, \\
\quad \{\, \langle\, \langle\, \text{TALLER}, \text{MS}\, \rangle, \text{TRUE}\, \rangle \\
\quad\quad \langle\, \langle\, \text{BELIEVES}, \text{M}\, \rangle, \langle \ldots \textit{nested view} \ldots \rangle\, \rangle \\
\quad\quad \langle\, \langle\, \text{MARY_P}, \text{M}\, \rangle, \text{TRUE}\, \rangle \\
\quad\quad \langle\, \langle\, \text{WENDY_P}, \text{W}\, \rangle, \text{TRUE}\, \rangle\, \} \\
\quad \rangle \{ \ldots \textit{some mapping} \ldots \}\, \rangle
\end{array}$$

Where 'MARY_P' is the functor associated with the constant '`mary`' by a 'constants' definition and, similarly, 'WENDY_P' is associated with '`wendy`'.

---

[2]That is, a single action rule or scenario definition

[3]There are of course an infinite number of views of this form, which one we assume is the denotation of the outline is unimportant, so long as each outline denotes a different view

§3.2.3 Outlines as Views

The mapping which forms one part of a view is defined to be between entities in the context which forms its other part and some other set '**e**' which is supplied by the surrounding context. In order to define this mapping we must determine what the 'surrounding' context is in each case. To do this we assume a function 'Parent_outline' which, for any ⟨outline⟩, gives another ⟨outline⟩ which defines the context with respect to which its associated view is to be defined. Most views will be represented by ⟨state outline⟩s which occur textually within others, in this case we make this textual nesting define the parent relationship:

For *child* a ⟨outline⟩ which occurs within braces after a ⟨modal term⟩ define

$$\begin{aligned}\text{Parent\_outline}(\mathit{child}) = \mathit{parent}&\\ \text{where}&\\ \mathit{parent}&\text{ is the outline containing}\\ \text{the }&\langle\text{modal term}\rangle\end{aligned}$$

In all other cases the relationship 'Parent_outline' will be defined by the larger structure of which the ⟨outline⟩ is a part, see section §4.

Given this function we can more fully define the view which is represented by a ⟨outline⟩ '*d*', it must be a member of $\text{Views}_{U_D}(\mathbf{e}_{c_v})$, where

$$v = [\![\text{Parent\_outline}(d)]\!]$$

, and its context must be of the correct form, as described in the previous section. We now need to be more specific about just which of the views which fit this outline will be the one represented. To do this we must define the mapping which is assumed between the entities in the view and the entities in its parent context.

For *d* a ⟨outline⟩ define

$$\begin{aligned}\text{Mapping\_to\_parent}(d) = \{\, \langle\, \mathit{parent\_e}, \mathit{child\_e}\,\rangle \mid&\\ \mathit{child\_e} \in \mathbf{e}_{\mathit{child\_c}} \wedge&\\ \mathit{parent\_e} \in \mathbf{e}_{\mathit{parent\_c}} \wedge&\\ \exists\, \langle\text{name}\rangle \text{ s.t. } \mathit{child\_e} = \text{Entity\_For}(\langle\text{name}\rangle, d) \wedge&\\ \mathit{parent\_e} = \text{Entity\_For}(\langle\text{name}\rangle, p)\,\}&\\ \text{where}&\\ p = \text{Parent\_outline}(d) \wedge&\end{aligned}$$

$$child\_c = c_{[\![d]\!]} \wedge$$
$$parent\_c = c_{[\![p]\!]}$$

That is, entities are mapped by pairing them with entities in the other view which were denoted by the same ⟨**name**⟩.

We will use the convention that the value 'FALSE' for the 'Parent_outline' of an outline means that the mapping should be constructed with respect to the 'basecontext' of the planning domain. Such outlines form the top level of the structures of views. In these cases the mapping will be given by

For d a ⟨**outline**⟩ s.t. Parent_outline(d) = FALSE define

$$\begin{aligned}\text{Mapping\_to\_parent}(d) = \{ \langle e, child\_e \rangle \mid\ & child\_e \in \mathbf{e}_{child\_c} \wedge \\ & e \in \mathbf{e}_{basecontext_D} \wedge \\ & \exists \text{`}\mathbf{name}\text{'} \text{ s.t. } child\_e = \text{Entity\_For}(\text{`}\mathbf{name}\text{'}, d) \wedge \\ & e = \text{Den}(\text{`}\mathbf{name}\text{'}) \}\end{aligned}$$

where

$$child\_c = c_{[\![d]\!]}$$

In this case we map entities in the child onto the entities which were associated with the same name by a 'constants' declaration. This will give no mapping for entities representing place holders since they are not declared and so do not occur in the 'Den' function of a domain. This is to ensure that each use of, for instance, a rule can interpret the place holders as it likes and not as the 'basecontext' directs. In effect it is as if all of the place holders in a 'root' ⟨**outline**⟩ of this kind were included in the list of 'quantified' place holders at the top.

To continue the example given earlier; the ⟨**outline**⟩

```
taller(mary, ?someone)
believes(mary) = {
       ~taller ( mary, wendy )
}
```

Would represent a view of the following form

$$\begin{aligned}&\langle \langle \{ \text{M}, \text{S}, \text{W} \}, \\ &\quad \{ \langle \langle \text{TALLER}, \text{M}, \text{S} \rangle, \text{TRUE} \rangle, \\ &\quad\ \langle \langle \text{BELIEVES}, \text{M} \rangle,\end{aligned}$$

$$
\begin{array}{l}
\langle\langle\{\mathrm{M_2W_2}\},\\
\quad\{\langle\langle \mathrm{TALLER}, \mathrm{M_2}, \mathrm{W_2}\rangle, \mathrm{FALSE}\rangle,\\
\quad\ \langle\langle \mathrm{MARY\_P}, \mathrm{M_2}\rangle, \mathrm{TRUE}\rangle,\\
\quad\ \langle\langle \mathrm{WENDY\_P}, \mathrm{W_2}\rangle, \mathrm{TRUE}\rangle\}\\
\quad\rangle\{\langle \mathrm{M}, \mathrm{M_2}\rangle, \langle \mathrm{W}, \mathrm{W_2}\rangle\}\\
\ \rangle\rangle\\
\langle\langle \mathrm{MARY\_P}, \mathrm{M}\rangle, \mathrm{TRUE}\rangle\\
\langle\langle \mathrm{WENDY\_P}, \mathrm{W}\rangle, \mathrm{TRUE}\rangle\\
\}\rangle\\
\{\ldots \textit{some mapping} \ldots\}\rangle
\end{array}
$$

Where the mapping for the outer view would depend on the context in which it is used.

§3.2.4 Place Holders For Views

The intended use of the syntax being described here is the definition of actions and rules in a domain description. In such a definition one often needs to express things which do not fit into the simple semantics given above. In particular, in order to define the audience models we developed in chapter 7 we need to define actions with place holders which range over sets of facts, for instance to say things like "If the antecedent of an inference is believed by an agent then that agent is able to perform action 'infer' resulting in the consequent of that action being believed". We call these place holders 'second order place holders', and use the syntax given in the last alternative in the syntax of ⟨**state outline**⟩s given earlier. That is, a place holder in a position where we would expect a term in the ⟨**outline**⟩.

A full semantics for second order place holders would require a substantial extension to the representation to allow views to contain place holders which can later be bound to sets of facts. This would give the representation as a whole a much more complex semantics as binding a place holder might change the denotation of a view substantially rather than just refining it. However for our present purposes we need not supply a full solution to this problem as the only time we shall need 'second order' place holders will be in action definitions and they will be instantiated on inserting an instance of the action into the plan. Thus the representation for these place holders can be quite simple relying on the planning algorithm to interpret them.

We choose to represent a view with an embedded place holder `?c` by assigning the special term ⟨PH, C⟩ the value true in the view. As we will see in appendix C the effect of this will be to allow the planning algorithm to match an arbitrary term to the view.

Another type of 'second order' place holder occurs when the ⟨state outline⟩ following a modal term is replaced by a ⟨place holder⟩ and that place holder is not associated with a view by some other use. In this case it was stated in section §3.2.2 that the value of the modal term is given as an entity from the domain. This kind of place holder can be seen to be a simple variable-over-views. It will be instantiated to an actual view when the action or rule containing it is inserted into the plan.

These two types of place holder are most useful when combined. For instance the following fragment might be part of an action —

```
believe(?anyone) = ?c
believed_by_someone = {
    ?c
    }
```

which states that the beliefs of any agent form a subset of the view which is the value of `believed_by_someone`. In the precondition of an action this would trigger attempts to support complex facts of the form

believed_by_someone, XXX

for each complex term 'XXX' in the beliefs of the agent bound to `?anyone`.

Similarly in the post condition of an action the above fragment would give the action side-effects to the effect that anything it supports as a belief of `?anyone` will also become part of the value of `believed_by_someone`.

§4 Scenarios, Actions and Rules

After the declarations of the functors, constants and agents which are to be part of a planning domain, there come definitions of the structured elements of the domain – 'scenarios', 'actions' and 'rules'. The syntax and semantics (in terms of views) of these definitions will be given in the following 3 subsections.

§4.1 Scenario Definitions

The system must be given a description of the initial state of the world when it is started. This is done by defining 'scenario's. The syntax is defined to allow multiple scenarios to be defined and referred to by name.

The syntax of a scenario definition is as follows

⟨scenario definition⟩ ⇒ 'scenario' ⟨name⟩ ⟨state outline⟩ ';'

This definition ensures that the set 'scenarios' in the planning domain currently being defined contains the pair

⟨ [[name]], [[state outline]] ⟩where
Parent_outline(⟨state outline⟩) = FALSE

That is, the described view is relative to the base context of the domain.

Once a scenario has been defined in this way Riple can be started with this as the initial state of the world by issuing a 'plan' command with the ⟨name⟩ as argument.

§4.2 Action Definitions

'Actions' are descriptions of the ways which are available to an agent in the domain to change the state of the world. See section §5.1.3 of chapter 5 for a discussion of what we mean by an action. The actions are described in three parts, first a description of the action itself which is given as a list of 'descriptive' terms, next a description of the way the world should look before the action is possible and finally a description of the way the world should look after the action is performed.

The syntax of an action definition is as follows —

⟨action definition⟩ ⇒ 'action' ['visible'] ⟨name⟩
⟨action outline⟩
'from'
⟨state outline⟩
'to'
⟨state outline⟩
'end'

The outlines are to be interpreted with

$$\text{Parent\_outline}(\langle \textsf{action outline} \rangle) = \langle \textsf{state outline} \rangle^2$$
$$\text{Parent\_outline}(\langle \textsf{state outline} \rangle^1) = \langle \textsf{state outline} \rangle^2$$
$$\text{Parent\_outline}(\langle \textsf{state outline} \rangle^2) = \text{FALSE}$$

That is, the description of the action and the description of the world before the action is performed are interpreted as being 'inside' the description of the world after the action. This is done so that the results of an action can be used as the key in looking up which actions can be used to perform a given operation and the description of the action and the preconditions can be interpreted in terms of the interpretation made of the results.

The meaning of of an ⟨action definition⟩ where '`visible`' is not present is a 3–tuple

$\langle\, desc, prec, postc \,\rangle$
where
$desc = [\![\textsf{action outline}]\!]$
$prec = [\![\textsf{state outline}]\!]^1$
$postc = [\![\textsf{state outline}]\!]^2$

Which is inserted into the set 'actions' of the planning domain being defined.

If '`visible`' is present the action is modified to ensure that all agents know that it has been performed —

$\langle\, desc, prec, postc \,\rangle$
where
$desc = \text{Make\_visible}([\![\textsf{action outline}]\!])$
$prec = \text{Make\_visible}([\![\textsf{state outline}]\!]^1)$
$postc = \text{Make\_visible}([\![\textsf{state outline}]\!]^2)$

Where Make_visible is defined as —

For $\mathbf{e} \subseteq \mathbf{entities}_U$ and $v \in \text{Views}_U(\mathbf{e})$ define

$\text{Make\_visible}(v) = v' \in \text{Views}_U(\mathbf{e})$
where
$\forall\, a \in \mathbf{agents}_D,$
$\text{Assigned\_Val}(v', t) = \text{Assigned\_Val}(v, t) \oplus v$

$$\begin{aligned} &\text{where } t = \langle\, \mathit{believe}, a' \,\rangle \\ &\quad \langle\, a, a' \,\rangle \in \mathrm{M}_v \\ &\quad \mathit{believe} = \mathrm{Den}(\texttt{'believe'}) \end{aligned}$$

An example of an ⟨action definition⟩ is

```
action put_on
    move(?block1, ?block2)
from
    clear(?block1)
    clear(?block2)
to
    on(?block1, ?block2)
end
```

Which, assuming the functors had been declared in the obvious manner, would create the tuple

$$\begin{aligned}
&\langle\langle\langle\{\mathrm{V}_1, \mathrm{V}_2\}, \\
&\quad \{\langle\langle \mathrm{MOVE}, \mathrm{V}_1, \mathrm{V}_2\rangle, \mathrm{TRUE}\rangle\} \\
&\quad \rangle, \\
&\quad \{\langle \mathrm{V}_5, \mathrm{V}_1\rangle, \langle \mathrm{V}_6, \mathrm{V}_2\rangle\} \\
&\quad \rangle, \\
&\langle\langle\{\mathrm{V}_3, \mathrm{V}_4\}, \\
&\quad \{\langle\langle \mathrm{CLEAR}, \mathrm{V}_3\rangle, \mathrm{TRUE}\rangle \\
&\quad\ \langle\langle \mathrm{CLEAR}, \mathrm{V}_4\rangle, \mathrm{TRUE}\rangle\} \\
&\quad \rangle, \\
&\quad \{\langle \mathrm{V}_5, \mathrm{V}_3\rangle, \langle \mathrm{V}_6, \mathrm{V}_4\rangle\} \\
&\quad \rangle, \\
&\langle\langle\{\mathrm{V}_5, \mathrm{V}_6\}, \\
&\quad \{\langle\langle \mathrm{ON}, \mathrm{V}_5, \mathrm{V}_6\rangle, \mathrm{TRUE}\rangle\} \\
&\quad \rangle, \\
&\quad \{\,\} \\
&\quad \rangle \\
&\rangle
\end{aligned}$$

Notice that the first two views have their entities described in terms of those in the third.

§4.3 Rule Definitions

'Rules' define constraints on the behaviour of agents, see section §5.3.1 of chapter 5. A rule consists of a condition which is used to recognise when a rule is applicable and a description of a desired state of the world which the agent must try to attain when the condition is satisfied. Rules are necessary to produce goals for the system to satisfy since a domain with no rules will never propose goals for its agents and hence will be totally static.

The syntax of a rule definition is

⟨rule definition⟩ ⇒ 'rule' ⟨name⟩
⟨state outline⟩
'gives'
⟨state outline⟩
'end'

The 'Parent_outline' definition for such a definition is as follows

Parent_outline(⟨state outline⟩^{2}) = ⟨state outline⟩^{1}
Parent_outline(⟨state outline⟩^{1}) = FALSE

That is, the set of goals is interpreted in terms of the condition. Once again this is to enable the parent view to be used as a key for lookup and the interpretation of it which is found to be applied to the set of goals.

The meaning of a ⟨rule definition⟩ is a pair —

⟨ [[state outline]]^1, [[state outline]]^2 ⟩

Which is inserted into the set 'rules' of the planning domain being described.

An example of a ⟨rule definition⟩ is

```
rule make_clear
    on(?block, myblock)
gives
    clear ( myblock )
end
```

This rule ensures that whenever something is put onto the block represented by the constant 'myblock', the system will be given a goal to clear 'myblock'. This rule would build a structure as follows.

$$
\begin{array}{l}
\langle\langle\{\,\mathrm{V}_1, \mathrm{MYBLOCK}_1\,\}, \\
\quad \{\langle\langle \mathrm{ON}, \mathrm{V}_1, \mathrm{MYBLOCK}_1\rangle, \mathrm{TRUE}\rangle, \\
\quad\ \langle\langle \mathrm{MYBLOCK\_P}, \mathrm{MYBLOCK}_1\rangle, \mathrm{TRUE}\rangle\,\} \\
\quad\rangle, \\
\quad \{\langle \mathrm{MYBLOCK}, \mathrm{MYBLOCK}_1\rangle\,\} \\
\quad\rangle, \langle\langle\{\,\mathrm{MYBLOCK}_2\,\}, \\
\qquad \{\langle\langle \mathrm{CLEAR}, \mathrm{MYBLOCK}_2\rangle, \mathrm{TRUE}\rangle, \\
\qquad\ \langle\langle \mathrm{MYBLOCK\_P}, \mathrm{MYBLOCK}_2\rangle, \mathrm{TRUE}\rangle \\
\qquad\ \}\rangle, \\
\quad \{\langle \mathrm{MYBLOCK}_1, \mathrm{MYBLOCK}_2\rangle\,\} \\
\quad\rangle \\
\rangle
\end{array}
$$

Assuming that '`myblock`' has been defined as a constant and that the entity 'MYBLOCK' in the basecontext has been made its denotation.

§5 Agents and Beliefs

So far in this description no mention has been made of any difference between the behaviour of those names which are declared to represent constants and those which are declared to represent agents. In most respects the two types of object are, indeed, treated identically in the syntax described in the previous section and the manipulations in the previous chapter, and it is only in the behaviour of the algorithms which manipulate the representations to produce the behaviour of the system that distinguish between them.

However there is one way in which the semantics of, say, a rule definition must be affected by the agents which exist in the domain being described, the interactions between agents and their beliefs.

Agents, in the model developed in chapter 5, have two distinguishing features —

Actions Any agent is assumed to be able to perform actions as defined in the domain.

Beliefs Each agent is assumed to have an independent set of beliefs, including beliefs about the beliefs of other agents (and so on recursively).

The first property is a result of the planning activity of the system and so has no effect on the semantics of the declarations and definitions described in this chapter. However the second must have an effect on the semantics of descriptions of the world. Beliefs are assumed to have the following properties (see section §5.4.1 of chapter 5) —

Ownership Every belief set will contain an entity which represents the agent whose beliefs it represents.

Reflexivity Any agent is assumed to have full knowledge of their own beliefs. That is, an agent's beliefs and their beliefs about their beliefs are indistinguishable.

Amongst other things, these properties ensure that an action whose stated effect is to make the agent performing it believe that a certain condition holds will be indistinguishable from one which is defined to actually make the condition hold. This type of identity, which may seem counter intuitive at first, is valid since the system under discussion here is not designed to describe the behaviour of an agent from the point of view of an external observer, rather it is designed to be that agent's behaviour.

The following subsections describe the modifications to the semantics given in sections §3, §2.2.2 and §2.2 which allow the beliefs of the agents to be represented and to possess this 'reflexivity' property.

§5.1 Agent Declarations, The 'Planner' Constant and The 'Believe' Functor

Agent declarations are of much the same form as constant declarations and have the much the same semantics, with the sole exception that they declare names of things to be inserted into the set $\mathbf{agents}_D$ rather than $\mathbf{constants}_D$ for the domain 'D'. The syntax of an agent declaration is —

⟨agent declaration⟩ ⇒ 'agents' ⟨name⟩* ';'

And the constraint it places on the domain is

For each ⟨name⟩ n in the declaration define

$$
\begin{aligned}
&\exists\, e \in \mathbf{e}_{basecontext_D} \text{ s.t.} \\
&\quad e \in \mathbf{agents}_D \wedge \\
&\quad \mathrm{Den}(n) = e \wedge \\
&\quad \mathrm{Is\_Pred}_D(e) = f \\
&\quad \text{where} \\
&\qquad f \in \mathbf{functors}_{U_D} \wedge \\
&\qquad \mathrm{Arity}_{U_D}(f) = 1 \wedge \\
&\qquad \mathrm{Ptype}_{U_D}(f) = \text{BOOLEAN}
\end{aligned}
$$

An agents beliefs about the beliefs of some agent (their own or those of some other agent) are represented by the value of a modal term with the unary modal functor '`believes`'. This functor is automatically declared to be in any domain. Also every domain contains an agent, '`planner`' representing the planning system itself. Every domain behaves as if it contains declarations of the form

```
functors
    modal
        believes/1
    ;

agents
    planner
    ;
```

Although it does no harm to explicitly include such a declaration for purposes of clarity.

Additionally every domain will contain, in its universe, a predicate 'ME_P' used, as described in the next section, to distinguish an agent's concept of itself.

§5.2 Agents and Outlines

Certain ⟨outline⟩s denote views which are to represent agents' beliefs about the state of the world. Such views must be structured so that the two properties of "ownership" and "reflexivity" described above hold.

In order to know how to structure each view we must distinguish ⟨outline⟩s which represent the beliefs of some agent from those which are used for other purposes. The following types of ⟨outline⟩ are defined to be "belief outlines":

- The two ⟨state outline⟩s which form a rule definition.

- The two ⟨**state outline**⟩s which form an action description.
- The ⟨**state outline**⟩ which forms the body of a scenario description.
- Any ⟨**state outline**⟩ which is given as the value of a term whose functor is '`believes`' in a belief outline.

§5.2.1 "Ownership" and Outlines

In a belief outline, the special term argument '`ME`' behaves as if it was a constant whose 'Is_pred' is 'ME_P' except that the entity which represents this "constant" is not bound to that representing it in the 'Parent_outline' of the belief outline. In any non-belief ⟨**outline**⟩, '`ME`' behaves like any other constant, that is it will be bound to the '`ME`' entity in its parent.

The effect of this will be to put in each belief context, and in any non-belief contexts which descend from it, a distinguished entity which has the properties we require for an agent's concept of itself and which can be referred to syntactically using the token '`ME`'.

The interpretation of the entity represented by 'ME' in the mapping of the view which a belief outline denotes depends on the type of belief outline.

- In the outlines forming the bodies of ⟨**rule definition**⟩s, ⟨**action definition**⟩s and ⟨**scenario definition**⟩s the "constant" '`ME`' just follows the normal rules for constructing the mapping described in section §3.2.3, that is they are related according to the 'Parent_outline' relationship.
- In outlines which are given as the value of 'believes' terms, the entity representing 'ME' is mapped to that which is the argument of the 'believes' functor.

§5.2.2 "Reflexivity" and Outlines

In the view, 'v', denoted by any belief outline the term

$$\langle [\![\text{believes}]\!], [\![\text{ME}]\!] \rangle$$

will have as its value a view whose context is c_v and whose mapping is the identity function.

§5.3 Summary of Agent Representation

The final result of all of the rules described in this section is to force the structures of views which the system uses to represent actions, rules and scenarios to conform to the structural constraints described in section §5.4.1 of chapter 5.

§6 Domain Descriptions for Riple

The implemented system, Riple accepts domain descriptions in the language defined in this chapter and from them builds structures to represent the domain which are analogues of the mathematical structures given here as the semantics of the language. The only difference between the language understood by the system and that given here is that, since Riple is built on top of pop11, it uses the pop11 lexical analyser. This means that ⟨name⟩'s may be any legal pop11 word (see [Barrett et al. 1985]), rather than just alpha-numeric sequences as described here.

Appendix C

Planning in a Multi-Agent Environment

§1 Introduction

Having introduced a representation and a syntax for defining planning domains together with its interpretation in terms of the representation, we are in a position to describe the planning algorithm.

In chapter 5 we presented this algorithm informally. In this chapter we cover much of the same ground in a more formal manner. The aim of this chapter is to present the model in sufficient detail to allow the reader to follow the discussion of text planning in chapter 7; for this reason we do not describe the very lowest levels of the system.

The planning model presented here is under-specified in a number of ways. In general we have tried to develop a fairly neutral, if limited, model of planning in a multi-agent domain and have tried to avoid building assumptions into the model. However when the model was realised as a computer program, some assumption had to be made in each of these under-specified areas. section §5 of this chapter describes the assumptions which were made and presents justifications for them.

§2 Plan Representation

As was described in §5.1.2 of chapter 5, we represent a plan as a collection of views each of which is a (partial) description of the state of the world at some stage of the execution of the plan. Each of these plan states are related to each other by links indicating the earliest and latest times at which the description is assumed to hold and these links may be either fixed or movable. Additionally, the plan must contain a description of each action which forms part of the planned sequence of events and it must note the relationships between the actions and states and the agent who is to perform each action and which agent is responsible for each state.

Thus we represent a plan 'P' (in a planning domain D whose universe U_D we denote by U) as a triple —

$$\langle\, \mathbf{states}_P, \mathbf{actions}_P, \text{Earliest}_P, \text{Latest}_P, \text{Before}_P, \text{After}_P, \text{Owner}_P \,\rangle$$

where

states$_P$ is a subset of Views$_U$($\mathbf{e}_{basecontext_D}$) each element of which is a 'belief view'; that is they must satisfy the conditions given in section §5.4.1 of chapter 5.

actions$_P$ is a subset of Views$_U$(**entities**$_U$).

Earliest$_P$ is a function from **states**$_P$ to $\{\text{FALSE}\} \cup (\mathbf{states}_P \times \{\text{TRUE}, \text{FALSE}\})$. The value of this function is FALSE for the start state of the plan. For any other state S, Earliest$_P(S)$ is a pair consisting of the state after which the argument state is assumed to start and a flag indicating if the link is movable.

Latest$_P$ is also a function from **states**$_P$ to $\{\text{FALSE}\} \cup (\mathbf{states}_P \times \{\text{TRUE}, \text{FALSE}\})$. Its interpretation is parallel to that of Earliest$_P$, it is FALSE for the final state in the plan.

Before$_P$ is a function from **actions**$_P$ to **states**$_P$ giving the state which forms the precondition of the action.

After$_P$ is a function from **actions**$_P$ to **states**$_P$ giving the state which forms the postcondition of the action.

Owner$_P$ is a function from $\mathbf{actions}_P \cup \mathbf{states}_P$ to $\mathbf{agents}_D$ giving for each action the agent who performs it and for each state the agent who is responsible for supporting the complex facts it asserts.

§3 Preliminary Definitions

First we define the notion of matching used in the rest of this chapter; two complex terms match (under a substitution ϕ) iff ϕ substitutes entities for place holders in the terms and the results are equal —

For $\mathbf{e} \subseteq \mathbf{entities}_U$ and $t_1, t_2 \in \text{Cterms}_U(\mathbf{e})$ define

$$t \sim_\phi t' \Leftrightarrow \phi :_{\text{part}} \mathbf{e}' \rightarrow \mathbf{e} \wedge$$
$$t/\phi = t'/\phi$$
where
$$\mathbf{e}' = \text{Terms}_U(\mathbf{e}) \cup \mathbf{e} \setminus \text{CoDom(Den)}$$

The expression $\mathbf{e} \setminus \text{CoDom(Den)}$ is used to discriminate between entities representing place holders and those representing constants; the latter will be the denotation of some name. This means that the above definition of matching is only useful for complex terms stated in terms of the entities of the base context of the domain. This is all that is needed and, in fact, all that is possible. Only by looking at the domain can the system determine which constants exist. We shall refer to substitutions which only affect place holders, such as those used in matching, as instantiations.

We also define two more types of matching which are central to the operation of the system. First, we must know how to match a complex term against a (belief) view; this is used to find support for a complex term.

For $\mathbf{e} \subset \mathbf{entities}_U$, $v \in \text{Views}_U(\mathbf{e})$ and $t \in \text{Cterms}_U(\mathbf{e})$ define

$$t \sim_\phi v \Leftrightarrow \exists\, t' \in \text{Cterms\_In}(v, \emptyset) \text{ s.t. } t \sim_\phi t'$$

Next we must be able to match an arbitrary view against a state in the plan. There are two versions of this, first we have a simple generalisation of the preceding match definitions.

For $\mathbf{e} \subset \mathbf{entities}_U$ and $v, v' \in \text{Views}_U(\mathbf{e})$ define

$$
\begin{aligned}
v \sim_\phi v' \Leftrightarrow & \forall t \in \text{Cterms\_In}(v, \emptyset), \\
& \exists t' \in \text{Cterms\_In}(v', \emptyset) \text{ s.t.} \\
& \quad t \sim_\phi t' \wedge \\
& \quad \text{Assigned\_Val}(v, t) = \text{Assigned\_Val}(v', t')
\end{aligned}
$$

We use the second matching definition to describe the conditions for applying a rule and, as was discussed in section §5.3.1 of chapter 5, we must ensure that this matching process does *not* bind any place holders in the plan. For this reason we shall use the asymmetric symbol $\succ$ to denote this form of matching. $v \succ_\phi v'$ means "all complex terms in v, when transformed using the substitution ϕ, match some term in v' which has the same value" —

For $\mathbf{e} \subset \mathbf{entities}_U$ and $v, v' \in \text{Views}_U(\mathbf{e})$ define

$$
\begin{aligned}
v \succ_\phi v' \Leftrightarrow & \forall t \in \text{Cterms\_In}(v, \emptyset), \\
& \exists t' \in \text{Cterms\_In}(v', \emptyset) \text{ s.t.} \\
& \quad \phi :_{\text{part}} \mathbf{e}' \rightarrow \mathbf{e} \wedge \\
& \quad t/\phi = t' \\
& \quad \text{where} \\
& \qquad \mathbf{e}' = \mathbf{e} \setminus \text{CoDom}(\text{Den}) \wedge \\
& \quad \text{Assigned\_Val}(v, t) = \text{Assigned\_Val}(v', t')
\end{aligned}
$$

The notion of a 'belief view' described in section §5.4.1 of chapter 5 is central to the planning process. The following definition allows us to talk about the belief views which are present at a state of the plan. We represent a belief view by the sequence of *believe* terms leading up to it. Recall that all the states in the plan are assumed to be belief views for the system itself and so the empty tuple of belief views is included to represent this.

For $v \in \mathbf{states}_P$ define

$$
\begin{aligned}
\text{Belief\_Views}(v) = \{ \, & \mathit{cterm} \in \text{Cterms\_In}(v, \emptyset) \mid \\
& \exists a_1..a_n \in \mathbf{agents}_D \text{ s.t. } \mathit{cterm} = \\
& \quad \langle \langle \mathit{believe}, a_1 \rangle, \langle \mathit{believe}, a_2 \rangle, \ldots \\
& \quad \langle \mathit{believe}, a_n \rangle \rangle \} \cup \{ \langle \, \rangle \} \\
& \text{where } \mathit{believe} = \text{Den}(\texttt{'believe'})
\end{aligned}
$$

A final useful notion is that of ordering. Two states are ordered if they can be linked by a sequence of Earliest_P and Latest_P relations.

For P a plan and $s, s' \in \mathbf{states}_P$ define

$$
\begin{aligned}
s \ll s' \Leftrightarrow\ & \text{Earliest}_P(s') = \langle\, s, x \,\rangle \vee \\
& \text{Latest}_P(s) = \langle\, s', x \,\rangle \vee \\
& \exists\, s'' \in \mathbf{states}_P \text{ s.t. } s \ll s'' \ll s' \\
& \text{where} \\
& \qquad x \in \{\, \text{TRUE}, \text{FALSE} \,\}
\end{aligned}
$$

§4 The Planning Cycle

As was described in section §5.1 of chapter 5, we view planning as an agenda based process where the system repeats the following sequence of actions —

- Find chores in the plan.
- Choose a chore to fix and find possible fixes.
- Apply the fix to obtain a new plan.

until there are no more chores in the plan.

This behaviour is potentially open ended since fixing the plan can introduce more chores.

In the following subsections we describe the chores associated with a plan, the fixes available for each chore and the method of performing those fixes. We do not discuss the method used to choose which chore to attempt and which fix to use, the details of these choices do not form part of the model we present here. Section §5 describes the choice methods used in the implemented program Riple.

§4.1 Finding Chores

The following types of chore can be present in a plan —

Support Every complex term which is assigned a value in a state in the plan must be supported; that is there must be sufficient constraints on the plan to ensure that the term will have the given value at the required time. The conditions which must hold for this to be true were outlined in section §5.2.3.1 of chapter 5; we give a more formal definition in section §4.1.1 below.

Motivation Every action in the plan must be motivated. Motivation has two parts —

Necessity The agent who is to perform the action must be responsible for an otherwise unsupported (complex) fact which the given action supports.

Optimality The action must be the preferred one for the motivating fact.

Rule If any belief view in any state in the plan matches the condition of a rule in the domain, there must be a later state which has a belief view in a parallel position which matches the goal set of the rule, see section §5.4.3 of chapter 5 for a full description of what is meant by parallel.

Conflict If the same complex term is given different values by two different states in the plan then those two states must be strictly ordered; that is there must be 'before' and 'after' links between the states which will prevent the states being true at the same time.

Binding Every place holder in a state in the plan must be replaced by a constant.

Reduction Every state in the plan must at some point be combined into the current state of the world. This is how 'time' moves forward in the system.

Execution Every action in the plan must be executed.

Below we examine each of these in more detail giving semi-formal definitions of the conditions which must hold for a chore to be induced by the plan.

§4.1.1 Support Chores

For any given plan the set of support chores which must be performed is simply the set of all complex terms given values by all of the states in the plan except those which already have support. The classes of complex terms which are deemed to be supported was described in section §5.2.3.1 of chapter 5. Here we simply formalise that description —

For P a plan define

$$\begin{aligned}\text{Unsupported}(P) = \bigcup_{v \in \text{Unsups}(P)} v \bowtie \{\, t \in \text{Cterms\_In}(v, \emptyset) \mid \\ \neg\exists\, s \in \textbf{states}_P \text{ s.t. } \text{Assigned\_Val}(s,t) = \\ \text{Assigned\_Val}(v,t) \\ \wedge\ \text{Supports}_P(s,v)\,\}\end{aligned}$$

where

$$\begin{aligned}\text{Unsups}(P) = \{\, s \in \textbf{states}_P \mid \\ \text{Before}_P(s) \neq \text{FALSE} \wedge \\ \neg\exists\, a \in \textbf{actions}_P \text{ s.t. } \text{After}_P(a) = s \\ \}\end{aligned}$$

$$\begin{aligned}\text{Supports}_P(s,s') = \text{Earliest}_P(s') = \langle\, s, \text{TRUE}\,\rangle \vee \\ \text{Latest}_P(s) = \langle\, s', \text{TRUE}\,\rangle\end{aligned}$$

Here, $\text{Unsups}(P)$ gives the set of states which are not intrinsically supported (that is, not the start of the plan or the result of an action) while $\text{Supports}_P(s,s')$ indicates that the relationship between s and s' is fixed, and so will provide support. We do not use a recursive version of Supports_P, this means that the supporting state must be immediately connected. This is not a problem since when we support states we insert a 'hold' state to tie supporter to supported.

§4.1.2 Rule Firing Chores

A plan contains a rule chore just in the case that there is a belief view in some state in the plan which matches the condition of the rule and no later state has a parallel belief view which matches the rules goal set under the same substitution.

For P a plan and D a planning domain define

$$\begin{aligned}\text{Fired}_D(P) = \{\, &\langle s, b, c, g, \phi \rangle \mid \\ &\langle c, g \rangle \in \textbf{rules}_D \wedge \\ &\exists\, s \in \textbf{states}_P, \\ &\quad b \in \text{Belief\_Views}(s) \text{ s.t.} \\ &\quad \text{Assigned\_Val}(s, b) = v \wedge c \succ_\phi v \wedge \\ &\quad \neg\exists\, s' \in \textbf{states}_P \text{ s.t.} \\ &\qquad \text{Assigned\_Val}(s', b) = v' \wedge g \succ_\phi v' \wedge \\ &\qquad s \ll s' \,\}\end{aligned}$$

§4.1.3 Conflict Chores

Two states are in conflict if they assign different values to the same complex fact and are not ordered in the plan.

For P a plan and $s, s' \in \textbf{states}_P$ define

$$\begin{aligned}\text{Conflicts}_P(s, s') = &\text{ if } s \ll s' \vee s' \ll s \text{ then} \\ &\qquad \emptyset \\ &\text{otherwise} \\ &\qquad \{\, t \in \text{Cterms\_In}(s, \emptyset) \mid \\ &\qquad\quad t \in \text{Cterms\_In}(s', \emptyset) \wedge \\ &\qquad\quad \text{Assigned\_Val}(s, t) \neq \text{Assigned\_Val}(s', t) \,\}\end{aligned}$$

§4.1.4 Motivation Chores

Determining which actions need to be motivated is fairly simple. Every action which is planned to be performed by someone other than the system itself must be motivated. The conditions under which an action is motivated were stated in section §4.1, 'Necessity' and 'Optimality', here we can present these conditions a little more formally.

An action is necessary if there is some complex term whose value has to be supported by the agent performing the action which would otherwise remain unsupported. This can be broken up into three parts — first the term must not have the required value when the action is performed; second the term must not be given the value by a later action and finally a state must exist later in the plan which

requires the term to have that value as support and which is the responsibility of the agent.

The first part of this definition can be stated as —

For P a plan, $s \in \mathbf{states}_P$ and $t \in \text{Cterms\_In}(s, \emptyset)$ define

$$\begin{aligned}\text{Antisup}_P(s,t) \Leftrightarrow \exists\, s' \in \mathbf{states}_P \text{ s.t. } &\text{Latest}_P(s') = \langle\, s, \textsc{true} \,\rangle \wedge \\ &t \in \text{Cterms\_In}(s', \emptyset) \\ &\text{Assigned\_Val}(s,t) \neq \text{Assigned\_Val}(s',t)\end{aligned}$$

This is essentially a special case of the support condition turned on its head. At first sight it might seem that we would need to encode the full support criterion in reverse to know that a complex term will be unsupported at some point, but it suffices to force the negation of the term to be present in the plan and so supported. This means that the system may do more work than is strictly necessary to ensure that the action is motivated — all that is strictly necessary is that the other agent not believe that the motivating fact will be true — but this is the best we can do within the present representation. To go further we would need to be able to represent and reason about negation of modal terms.

The second and third conditions are simple to formalise —

For P a plan, $s \in \mathbf{states}_P$, $s' \in \mathbf{states}_P$ and $t \in \text{Cterms\_In}(s, \emptyset)$ define

$$\begin{aligned}\text{Needed}_P(s,t,s') \Leftrightarrow &t \in \text{Cterms\_In}(s, \emptyset) \\ &t \in \text{Cterms\_In}(s', \emptyset) \\ &\text{Assigned\_Val}(s,t) = \text{Assigned\_Val}(s',t) \wedge \\ &\neg\exists\, s'' \in \mathbf{states}_P \text{ s.t. } s \ll s'' \ll s' \\ &\quad \text{Assigned\_Val}(s'',t) \neq \text{Assigned\_Val}(s',t)\end{aligned}$$

The final remaining problem is to formalise the notion of "Optimality". However the structures we have so far defined do not contain the information necessary to do this, and indeed a full definition would require a complete knowledge of the planning behaviour of all other agents. Since this is impossible we shall rely on the assumption of 'Agent Similarity' introduced in chapter 5 to give us the leverage we need. We assume that the system has heuristics which determine which action is most suitable for supporting a given complex term and the agent similarity assumption will allow us to assume that other agents will use functionally similar heuristics. The problem of "Optimality" is now reduced to that of ensuring that

the action being motivated is the first candidate suggested by the heuristics to support the motivating fact. Our definition of "Optimality" then is intimately related to the assumed behaviour of the agents in the domain. The details of the assumptions made in the concrete implementation of this model are described in section §5; for now it need only be noted that an action is in need of motivation iff it is not the first suggested fix to the unsupported fact which is being used to fulfill the "necessity" condition of motivation. For the present we assume that the following function reflects the heuristics used in the system —

For P a plan , $s \in \mathbf{states}_P$ and $t \in \text{Cterms\_In}(s, \emptyset)$ define

$$\text{Actions\_for}(P, s, t) = \langle \langle a, \phi \rangle \mid a/\phi \text{ is suggested by the system's heuristics as a way of supporting } t \text{ in } s \rangle$$

Now we can pull together all the conditions to define what we mean by motivation.

For P a plan, $a \in \mathbf{actions}_P$ and $agt \in \mathbf{agents}_D$ define

$$\begin{aligned}\text{Motivated}(a, agt, P) \Leftrightarrow \\ &\exists t \in \text{Cterms\_In}(e), t', s \in \mathbf{states}_P \text{ s.t.} \\ &\quad t = \langle \mathit{believe}, agt \rangle \bowtie t' \wedge \\ &\quad \text{Antisup}_P(e, t) \wedge \\ &\quad \text{Needed}_P(e, t, s) \wedge \\ &\quad \text{Actions\_for}(P, e, t) = \langle a', \phi \rangle \bowtie \mathit{rest} \wedge \\ &\quad a = a'/\phi \\ &\text{where} \\ &\qquad e = \text{After}_P(a) \\ &\qquad \mathit{believe} = \text{Den}(\text{`}\texttt{believe}\text{'})\end{aligned}$$

We demand that the complex term which is motivating the action be of the form $\langle\langle \mathit{believe}, agt\rangle \ldots\rangle$ to ensure that it is actually known to the agent being motivated.

§4.1.5 Binding Chores

The binding chores in a plan are simply the place holders present in the states.

For P a plan define

$$\text{Bindings}(P) = \{\, e \mid \langle e, e' \rangle \in M_v \wedge e \notin \text{Dom(Den)} \text{ where } v \in \mathbf{states}_P \cup \mathbf{actions}_P \,\}$$

§4.1.6 Reduction Chores

Section §5.2.3.4 of chapter 5 describes conditions under which plan state is a candidate for merging with the current state of the world. Three of the four conditions are special cases of chores described above (all complex terms supported, no conflicts and all place holders bound) and so we shall not repeat the definitions here. The only remaining condition is that the state be at the start of the plan, we describe this formally as follows —

For P a plan and $s \in \mathbf{states}_P$ define

$$\text{Starts}(s, P) \Leftrightarrow \text{Earliest}_P(s) = \langle s', x \rangle \wedge \text{Earliest}_P(s') = \text{FALSE} \text{ where } x \in \{\, \text{TRUE}, \text{FALSE} \,\}$$

§4.1.7 Execution Chores

The execution chores which need to be performed for a given plan can be found simply by looking for actions whose precondition state is the current state of the world.

For P a plan define

$$\text{Executable}(P) = \{\, a \in \mathbf{actions}_P \mid \text{Before}_P(a) = s \wedge \text{Earliest}_P(s) = \text{FALSE} \,\}$$

§4.2 Finding Possible Fixes

Now that we have presented the possible chores which the system might find to be done in a plan, we must define the allowable fixes. In general these are closely related to the chore definitions.

Once again we leave the details of how a fix is selected from the possibilities found unspecified. The choices made in implementing Riple are described in section §5.

§4.2.1 Fixes for Support Chores

There are two types of fix possible for chores involving providing support. Either a hold state can be introduced from an earlier point in the plan where the required complex fact is asserted, or a new action can be introduced into the plan to provide support.

Fixes of the first type are defined by —

For P a plan, $s \in \mathbf{states}_P$ and $t \in \text{Cterms\_In}(s, \emptyset)$ define

$$\begin{aligned}\text{Hold\_support\_fixes}(P, s, t) = \{ \langle s, t, s', \phi \rangle \mid \\ s' \not\gg s \wedge \\ \exists t' \in \text{Cterms\_In}(s', \emptyset), t'' \text{ s.t.} \\ t \sim_\phi t'' \wedge \\ t' \sim_\phi t'' \wedge \\ \text{Assigned\_Val}(s/\phi, t'') = \text{Assigned\_Val}(s'/\phi, t'') \\ \}\end{aligned}$$

Notice that the condition is that the supporting state should not be after the state to be supported; the fix can impose an ordering constraint on the plan. Also notice that the fix can bind place holders, the slightly roundabout way of expressing this in the definition is necessary to ensure that the substitution chosen binds only place holders.

The second type of fix is slightly more complex. We need to allow actions by any agent to be inserted into the plan and so we need to phrase the definition in terms of actions by an arbitrary agent. Also we must be careful to allow for disagreements about object identity.

First an auxiliary definition. We know that an (instance of an) action, performed by a given agent will have will have a given effect if executed in a given state if that agent will expect the results to be one state and the planner's view of that result is the desired effect.

For P a plan, $p \in \mathbf{states}_P$, $agt \in \mathbf{agents}_D$ and $a \in \mathbf{actions}_D$ define

$$\begin{aligned}\text{Action\_by}(agt, a, p) = \langle d, p, e \rangle \\ \text{where} \\ \text{Assigned\_Val}(p, \langle \mathit{believe}, agt \rangle) = p' \wedge\end{aligned}$$

$$
\begin{array}{l}
prec_a \sim_\phi p' \wedge \\
e' = p' \odot postc_a/\phi \wedge \\
d' = p' \odot desc_a/\phi \wedge \\
d \in \text{Views}_U(\mathbf{e}_{basecontext_D}) \\
\text{s.t.} \\
\qquad \forall t \in \text{Cterms\_In}(d', \emptyset), \\
\qquad\quad \text{Assigned\_Val}(d, t) = \text{Assigned\_Val}(d', t) \wedge \\
\qquad \text{Assigned\_Val}(d, \langle\, believe, agt \,\rangle) = d' \\
e \in \text{Views}_U(\mathbf{e}_{basecontext_D}) \\
\text{s.t.} \\
\qquad \forall t \in \text{Cterms\_In}(e', \emptyset), \\
\qquad\quad \text{Assigned\_Val}(e, t) = \text{Assigned\_Val}(e', t) \wedge \\
\qquad \text{Assigned\_Val}(e, \langle\, believe, agt \,\rangle) = e' \\
\text{if there is no such } \phi \text{ then UNDEFINED}
\end{array}
$$

This defines the result of the action to be a state represented by a view constructed as follows —

- Finds which instance of the actions preconditions matches the agent's beliefs in the given initial state (p'/ϕ).
- The postcondition from the system's point of view is the postcondition of the same instance of the action (e') viewed through the mapping of the precondition.
- In addition the postcondition describes what the agent will expect to happen.

The construction of the description of the action is similar.

Notice that in the special case of actions performed by the system, the above definition just reduces to instantiation. The view p' will have an identity mapping since it is the system's view of its own beliefs.

With this definition to hand we can define the set of possible action insertions which will provide a fix for a support chore —

For P a plan, $s \in \mathbf{states}_P$ and $t \in \text{Cterms\_In}(s, \emptyset)$ define

$$
\begin{array}{l}
\text{Action\_support\_fixes}(P, s, t) = \{\, \langle\, s, t, \phi, d, p, e, agt \,\rangle \mid \\
\qquad\qquad\qquad\qquad \exists\, a \in \mathbf{actions}_D \text{ s.t.}
\end{array}
$$

$$\begin{aligned}
&\text{Action_by}(agt, a, p) = \langle\, d, p, e \,\rangle \wedge \\
&\exists\, t' \in \text{Cterms_In}(e, \emptyset), t'' \text{ s.t.} \\
&\quad t \sim_\phi t'' \wedge \\
&\quad t' \sim_\phi t'' \wedge \\
&\quad \text{Assigned_Val}(e/\phi, t'') = \\
&\qquad\qquad \text{Assigned_Val}(s'/\phi, t'') \\
&\}
\end{aligned}$$

That is to say, the set of fixes is the set of possible action/agent/instantiation triples which will give the complex term to be supported as an effect.

§4.2.2 Fixes for Rule Firing Chores

Rule firing is much simpler than support. In finding a chore the system has identified a state, a chain of belief terms, an instantiation and a rule. There is only one possible fix, to put in a state of the correct type.

For P a plan, $s \in \mathbf{states}_P$, $v \in \text{Views}_U(\mathbf{e}_{basecontext_D})$ and ϕ a an instantiation define

$$\text{Rule\_fixes}(P, s, v, \phi) = \{\, \langle\, s, v/\phi \,\rangle \,\}$$

§4.2.3 Fixes for Conflict Chores

To fix a conflict, the planner must reorder the plan. In section §5.2.3.2 of chapter 5 the possible ways of doing this were discussed. The only complication is the necessity to allow for fixed links when moving the end points of a state. The following definitions just describe the link following process —

For P a plan and $s \in \mathbf{states}_P$ define

$$\begin{aligned}
\text{Far_end}(P, s) = &\text{if Latest}_P(s) = \langle\, s', \text{TRUE} \,\rangle \text{ then} \\
&\quad \text{Far_end}(P, s') \\
&\text{otherwise if Latest}_P(s) = \langle\, s', \text{FALSE} \,\rangle \text{ then} \\
&\quad s' \\
&\text{otherwise} \\
&\quad \text{UNDEFINED}
\end{aligned}$$

For P a plan and $s \in \mathbf{states}_P$ define

$$\begin{aligned}\text{Near\_end}(P, s) = &\text{if Earliest}_P(s) = \langle\, s', \text{TRUE}\,\rangle \text{ then}\\ &\quad \text{Near\_end}(P, s')\\ &\text{otherwise if Earliest}_P(s) = \langle\, s', \text{FALSE}\,\rangle \text{ then}\\ &\quad s'\\ &\text{otherwise}\\ &\quad \text{UNDEFINED}\end{aligned}$$

Now we can define the fixes. Exactly which fixes are viable depends on the ordering of the end points of the regions.

For P a plan and $s, s' \in \mathbf{states}_P$ define

$$\begin{aligned}\text{Conflict\_fixes}(P, s, s') = \{\, \langle\, x, w, y\,\rangle \mid\ & (b \not\ll b' \wedge\\ & x = \text{Far\_end}(P, e) \wedge y = b' \wedge w = \text{END}) \vee\\ & (e \not\gg e' \wedge\\ & x = \text{Near\_end}(P, b) \wedge y = e' \wedge w = \text{START}) \vee\\ & (b' \not\ll b \wedge\\ & x = \text{Far\_end}(P, e') \wedge y = b \wedge w = \text{END}) \vee\\ & (e' \not\gg e \wedge\\ & x = \text{Near\_end}(P, b') \wedge y = e \wedge w = \text{START})\,\}\\ \text{where}\\ & b = \text{Earliest}_P(s) \wedge\\ & b' = \text{Earliest}_P(s') \wedge\\ & e = \text{Latest}_P(s) \wedge\\ & e' = \text{Latest}_P(s')\end{aligned}$$

§4.2.4 Fixes for Motivation Chores

A first pass at finding Possible fixes for motivation chores might be to simply list those complex terms which are in the effects of the action to be motivated. This will give possible motivating facts.

For P a plan$agt \in \mathbf{agents}_D$ and $s \in \mathbf{states}_P$ define

$$\begin{aligned}\text{Motivators}(P, agt, s) = \{\, & t \in \text{Cterms\_In}(s, \emptyset) \mid\\ & t = \langle\, \mathit{believe}, agt\,\rangle \bowtie \langle \ldots \rangle\,\}\end{aligned}$$

However that is only part of the solution. We must also decide how to make the motivating fact unsupported so that the agent will indeed need to perform the action. There are three possibilities for this —

- The fact might already be part of a state in the plan which is the responsibility of the agent to be motivated and which is after the action.
- There might be an action which the agent could be made to perform which would have the motivating fact in its precondition, thus leaving it unsupported.
- There might be a rule which could be fired (on the agent's beliefs which would introduce the motivating fact.

The first case is obviously fairly straight forward —

For P a plan, $s \in \mathbf{states}_P$ and $t \in \text{Cterms\_In}(s, \emptyset)$ define

$$\text{Existing\_motivators}(P, s, t) = \{ \langle s', \phi \rangle \mid s' \not\gg s \wedge \exists t' \in \text{Cterms\_In}(s', \emptyset) \text{ s.t. } t \sim_\phi t' \}$$

Selecting an action which will provide motivation by having a given complex fact in its precondition is entirely parallel to the previously discussed case of selecting an action which will provide support by having a given fact in its post condition. For this reason we will omit the detailed definition from this discussion.

Selecting a rule which will have a given effect is also similar to providing support with an action and the details of this too will be omitted.

Notice that if an action is inserted as part of the motivation of another action, this action too will need to be motivated. The mechanism which decides which fix must be able to break possible cycles of action motivation chores which could be set up this way.

§4.2.5 Fixes for Binding Chores

The possibilities for binding place holders are simply the constants in the domain —

For P a plan and $x \in \mathbf{entities}_U$ define

$$\text{Binding\_fixes}(P, x) = \mathbf{e}_{basecontext_D} \setminus \text{Dom(Den)}$$

§4.2.6 Fixes for Reduction and Execution Chores

The reduction and execution chores is trivial to fix. In each case there is only one possibility, the state is merged or the action executed.

§4.3 Applying Fixes

Once chores have been determined and fixes found some, as yet unspecified process (see section §5) must select a fix to be applied to the plan. When this choice has been made the information provided by the fix must be used to modify the plan minimally to fix the problem. This section describes the plan fixing mechanisms.

Many of the fixes are similar, and so we start this section by describing the common mechanisms.

§4.3.1 Simple Plan Manipulations

The basic manipulations which must be performed on the plan are:

- Place holder instantiation.
- End point movement.
- State insertion.

§4.3.1.1 Place Holder Instantiation

Given an instantiation ϕ it is simple to apply it to all the plan as follows —

For P a plan and ϕ an instantiation define

$$\begin{aligned}\text{Instantiate\_plan}(P, \phi) = \langle\, &\mathbf{newst}, \mathbf{newac}, \text{Earliest}_P, \text{Latest}_P, \\ &\text{Before}_P, \text{After}_P, \text{Owner}_P \,\rangle \\ \text{where} \\ &\mathbf{newst} = \{\, s/\phi \mid s \in \mathbf{states}_P \,\} \\ &\mathbf{newac} = \{\, a/\phi \mid a \in \mathbf{actions}_P \,\}\end{aligned}$$

The place holder binding chore is obviously a special case of this where the instantiation is a singleton function.

§4.3.1.2 End Point Movement

State end points are moved in the obvious manner, by redefining the Earliest_P and Latest_P functions of the plan. The fix determination rules are responsible for ensuring that the movement is legal, that is that the end of a state is never before its beginning.

§4.3.1.3 State Insertion

Most state insertion is just as simple as the preceding manipulations, but there is one case where some non-trivial work must be done. It was stated in section §2 that all the states in a plan must be belief views for the planner itself. This means that they must conform to the structural constraints given in section §5.4.1 of chapter 5.

Because these constraints are based on the number of agents in the domain, and because this never changes, most of this work can be done when the domain is defined rather than while planning. However when the planning algorithm creates states "on the fly", rather than by instantiating rules, actions and scenarios, it must expand them to contain the self entities and the reflexive belief terms which the constraints require.

Finally it must be remembered that responsibility must be assigned for each inserted state.

§4.3.2 Simple Fixes

Some of the fix types are relatively simple, consisting of simple manipulations of the plan. These fixes are described together in this section.

'Hold' supporting fixes If an unsupported fact is to be supported by inserting a hold state then all tat need be done is to ensure that the new state is a legal belief view, as described above, and make the earliest and latest links of the state fixed to the point where the supported fact is asserted and where it is required respectively. Responsibility for hold states goes to the agent whose unsupported fact was being fixed.

Rule Firings Similarly, rule firing fixes are applied by just inserting the new state with its (movable) earliest link pointing to the state which fired it and its (fixed) end state pointing to the end of the plan. The new state is the responsibility of the agent responsible for the state which triggered the rule.

Conflict Resolution Conflicts are resolved simply by moving the indicated link.

Reduction Reduction is performed by simply merging the state to be reduced with the state of the world. The result is that of the '$\oplus$' operator defined in chapter A. All links in the plan which used to be set to the state which was reduced are made to point to the state of the world.

The remaining fix types are more interesting and are discussed in the following sections.

§4.3.3 Inserting Supporting Actions

When an action is inserted to provide support the states representing the pre- and post-conditions of the action are inserted with the latest link of the precondition fixed to the post condition and the earliest link of the post condition fixed to the precondition. The remaining two links are set movably to the start and and of the plan.

A hold state must then be inserted into the plan between the effects of the action and the unsupported fact. This ensures that the support can not be remover by later planning.

All of the newly inserted states are made the responsibility of the agent whose plan lacked support or motivation.

§4.3.4 Motivation

The simplest case of motivation is that where the motivating fact is already present in the plan. In this case all that need be done is to ensure that it is negated *before* the action needing motivation. This can be done by inserting a state consisting of just the negation of the motivating fact with its end point fixed to the effects of

the unmotivated action. The start point of this new state is movable and is just set to the start of the plan. We call this news state the 'motivation constraint'.

If motivation is to be provided by a rule being fired then in addition to the motivation constraint, correctly instantiated versions of the pattern and goal views of the rule are inserted with start and end links set to form a chain start of plan to pattern to goals to end of plan. Only the goal to end of plan link is fixed. The goals state is marked as the responsibility of the agent being motivated, but the pattern state is the responsibility of the planner.

When motivation is to be provided by an action, the action is inserted as normal. Its states are marked as the responsibility of the agent being motivated.

Once the motivating state and the motivation constraint are in place the remaining clause of the definition of motivation must be obeyed. This means that the system must be sure that the newly motivated action will be the one which is considered most suitable by the agent who is to perform it. How this can be done is obviously very dependent on the detailed behaviour of the agents involved.

For this discussion we shall just assume, as we did in chapter 5 that the heuristics provide a total order on the set of actions which might support the motivating fact (this is the function Actions_for which was referenced by the definition of the motivation chores in section §4.1.4).

When applying motivation fixes the system must use the action selection heuristics to get a list of actions, then it must make each of the possibilities in turn impossible by negating one of its preconditions until the action being motivated is at the front of the list.

The precondition negation can be simply achieved by inserting into the plan a state consisting of just the negation of one of the preconditions of the action to be blocked. The 'latest' link of this state is fixed to the effect of the action being motivated, its earliest link movable and initially set to the start of the plan. It is made the responsibility of the planner. This will cause the planner to ensure that the blocked precondition can not be made true until after the motivated action is performed.

§4.3.5 Action Execution

The main effect of executing an action, from the planner's point of view, is to change the initial state of the plan. As was described in section §5.2.3.5 of chap-

ter 5, the actual way this effects the outside world is not important to the planning algorithm. After the action has been executed the post-condition of the action is deleted form the plan and all links which formally pointed to it are made to point to the initial state. The post conditions are *not* merged into the state of the world, since that would be to assume that the action behaved as expected. If performing the action did not have the effects which the system expected, the plan will probably have some unsupported facts which were previously supported by the postcondition of the executed action. This will cause some new planning activity, giving a very simple form of execution monitoring.

§5 The Riple System

There are a number of places in which the model described in section §4 is underspecified, this reflects a methodological decision to develop a model which was, as far as possible, unconstrained by implementation issues so that the problems which arise from the basic model are easy to separate from problems with the actual program Riple. In this section we examine some of the assumptions made by Riple to flesh out the model into a working program.

§5.1 The Agenda

One very important question not addressed by the basic model is that of how one should decide which chore to work on at any time. There are many possible schemes, but to keep the program simple and easy to understand it was decided to use a simple multiple-agenda system. Each chore type has its own agenda where chores are placed in the order in which they are detected. Each agenda is given a priority and the system simply takes the first chore from the highest priority non-empty agenda. The program was written to allow the priorities to be changed, however in practice it was found that it was best to use the following priority order (from most urgent to least) —

1. Execute action.

2. Reduce.

3. Eliminate Conflict.

4. Find Support.

5. Motivate Action.

6. Fire Rule.

7. Bind Place Holder.

Although only the extremes of the order are really vital (making the system too keen to bind place holders, for instance, is a big loss since it loses much of the advantage of 'least commitment' search).

§5.2 Choice of Fix

Within each chore type, the presentation of fix types in section §4 was done in order of complexity and this is the order used in the program. Thus the system will attempt to insert a hold state in preference to introducing a new action to provide support.

In general, within one category, the fixes are tried in a totally arbitrary order. However the selection of which action to insert into a plan was found to be important for reasonable performance. Because of this, two heuristics were built into the coded equivalent of the Actions_for function.

1. Prefer actions whose postcondition contains a maximum of terms which are similar to those already in the plan. The exact measure of 'similarity' is rather ad-hoc, based on the number of functors and constants in common.

2. Prefer actions which were defined first. That is those which are near the top of the file defining the domain. This gives the person defining the domain some control over the priority assigned to actions.

Another important heuristic is to prefer actions to be performed by the system to those to be performed by other agents. This prevents much wasted time trying to motivate other agents to do things which the system could easily achieve.

§5.3 Recovery from Mistakes

It was mentioned in chapter 5 that a real planning system needs to be able to undo bad choices and try something else. In the implemented program this is achieved

by simple chronological backtracking. This is limited by the fact that it is not allowed to backtrack over action executions. It is probably that effort invested in a more intelligent search strategy would pay off in much better performance since the system spends much of its time in search.

§5.4 Tracking Changes

A final question which must be addressed in creating a working implementation is that of data structures and methods for doing the manipulations of the plan defined by the model. In accordance with the long term goal of 'self description' some effort was made to limit the number of extra structures and mechanisms used in the program. The plan is represented as a structure of views, not just at the level of states but also at the level of links and constraints. This was done by extending the representation slightly to allow views to be parts of terms. At present we have no formal model of the semantics of this extension and so it was not included in chapter A. However it seems well behaved and future work should produce a clean version, bringing self description one step closer.

As a representation, views have a very big advantage. The presence of the mapping in every structure helps with the management of tasks such as place holder binding and link movement. For instance, when a state is 'reduced' and merged with the initial state of the plan, the links which pointed to it automatically move to point to the initial state.

§6 Conclusion

In this chapter we have presented the details of an abstract model of planning in a multi-agent environment. Although this model is a long way from being a practical system for general planning work, it is designed to be as non-committal as possible in those areas where we have no strong theory to guide it, and hopefully much of the discussion in the next chapter about planning text will carry over into more complete multi-agent planners as these become available.

Finally the assumptions and short cuts of the present implementation of this model were described.

References

Allen, J. (1984a). Towards a general theory of action and time. *Artificial Intelligence*, 23:123–154.

Allen, J. F. (1984b). Towards a general theory of action and time. *Artificial Intelligence*, 23:123–154.

Appelt, D. E. (1982). Planning natural language utterances to satisfy multiple goals. Technical Report 259, AI Center, SRI.

Appelt, D. E. (1983). Telegram: a grammar formalism for language planning. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pages 595–599.

Appelt, D. E. (1988). Planning natural-language referring expressions. In McDonald, D. D. and Bolc, L., editors, *Natural Language Generation Systems*, chapter 3. Springer-Verlag, 69–97.

Austin, J. L. (1962). *How to Do Things With Words.* Oxford University Press, Oxford, second edition.

Barrett, R., Ramsay, A., and Sloman, A. (1985). *Pop11: A Practical Language for Artificial Intelligence.* Ellis Horwood, Chichester.

Bartle, R. (1984). Cross-level planning. Technical Report CSCM-16, Colchester Cognitive Studies Centre, University of Essex.

Barwise, J. and Perry, J. (1983). *Situations and Attitudes.* MIT press, Cambridge, Massachusetts.

Brown, G. and Yule, G. (1983). *Discourse Analysis.* Cambridge Textbooks in Linguistics. Cambridge University Press, Cambridge.

Cammarata, S., McArthur, D., and Steeb, R. (1983). Strategies of cooperation in distributed problem solving. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pages 767–770.

Chapman, D. (1987). Planning for conjunctive goals. *Artificial Intelligence*, 32:333–377.

Cheeseman, P. (1983). A representation of time for planning. Tech. note 278, SRI International, Menlo Park, CA.

Cohen, P. R. (1978). On knowing what to say:planning speech acts. Technical report 118, Department of Computer Science, University of Toronto.

Cohen, P. R. and Levesque, H. J. (1985). Speech acts and rationality. In *Proceedings of the ACL Conference*, pages 49–59.

Dale, R. (1986). The pronominalization decision in language generation. Research Paper 276, Department of Artificial Intelligence, University of Edinburgh.

Dale, R. (1989). *Generating referring expressions in a domain of objects and processes.* PhD thesis, University of Edinburgh.

Davey, A. (1978). *Discourse Production.* Edinburgh University Press, Edinburgh.

Dean, T. and McDermott, D. (1987). Temporal database management. *Artificial Intelligence*, 32:1–55.

DeKleer, J. (1986). An assumption based truth maintenance system. *Artificial Intelligence*, 28:127–162.

Doyle, J. (1979). A truth maintenance system. *Artificial Intelligence*, 12:231–272.

Fagin, R. and Halpern, J. Y. (1988). Belief, awareness and limited reasoning. *Artificial Intelligence*, 34:39–76.

Fauconnier, G. (1985). *Mental Spaces: Aspects of Meaning Construction in Natural Language.* MIT press, Cambridge, Massachusetts.

Fikes, R. E. and Nilsson, N. (1971). Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208.

Gazdar, G. (1979). *Pragmatics: Implicature, Presupposition and Logical Form.* Academic Press, New York.

Genesereth, M. R. and Nilsson, N. J. (1988). *Logical Foundations of Artificial Intelligence.* Morgan Kaufmann, Palo Alto.

Georgeff, M. P. (1987). Planning. *Annual Reviews in Computer Science*, 2:359–400.

Ginsberg, M. and Smith, D. (1987). Reasoning about action 1: A possible worlds approach. In Brown, F., editor, *The Frame Problem in Artificial Intelligence.* Morgan Kaufmann, Los Altos Ca.

Golding, W. (1980). *Rites of Passage.* Faber, London.

Green, C. (1969). Application of theorem proving to problem solving. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 741–747.

Grice, P. (1975). *Logic and Conversation.* Academic Press, New York.

Grosz, B. (1977). The representation and use of focus in a system for understanding dialogs. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, pages 67–76.

Grosz, B. J. and Sidner, C. L. (1986). Attention, intentions and the structure of discourse. *Computational Linguistics*, 12(3):175–204.

Hopkins, C. (1989). Meta-level planning for multi-agent cooperation. In Cohn, A. G., editor, *Proceedings of AISB 89*, pages 185–190. Society for the Study of Artificial Intelligence and Simulation of Behaviour, Morgan Kaufman.

Hovy, E. D. (1988). Planning coherent multisentential text. Reprint Series ISI/RS-88-208, Information Sciences Institute, University of Southern California, Marina del Rey.

Hovy, E. H. (1985). Integrating text planning and production in generation. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, volume 2, pages 848–851.

Hovy, E. H. and McCoy, K. F. (1989). Focusing your rst: A step towards generating coherent multisentential text. In *Conference of the Cognitive Science Society.*

Hughes, G. E. and Cresswell, M. J. (1968). *An Introduction to Modal Logic.* Methuen, London.

Jacobs, P. S. (1985). Phred: A generator for natural language interfaces. Technical Report UCB/CSD 85/198, Computer Science Division, UCB.

Kamp, H. (1984). A theory of truth and semantic representation. In Groenendijk, J., Janssen, T. M., and Stokhof, M., editors, *Truth Interpretation and Information*, pages 1–41, Dordrecht. Foris Publications.

Kay, M. (1986). Parsing in functional unification grammar. In Grosz, B. J., Jones, K. S., and Webber, B. L., editors, *Readings in Natural Language Processing*, pages 125–138. Morgan Kaufman.

Konolige, K. (1982). A first order formalisation of knowledge and action for a multi-agent planning system. In Hayes, J. E., Michie, D., and Pao, Y.-H., editors, *Machine Intelligence 10*, chapter 2, pages 41–72. Ellis Horwood, Chichester.

Kripke, S. A. (1963). Semantical considerations on modal logic. *Acts Philosophica Fennica*, 16:83–94.

Kripke, S. A. (1972). Naming and necessity. In Davidson, D. and Harman, G., editors, *Semantics of Natural Language*, pages 253–355. Reidel, Dordrecht.

Levesque, H. J. (1984). A logic of implicit and explicit belief. In *Proceedings of the 4th National Conference on Artificial Intelligence*, pages 198–202, Los Altos, Calif. American Association for Artificial Intelligence, William Kaufmann.

Mann, W. (1983). An overview of the penman text generation system. Technical Report RR-83-114, USC/ISI.

Mann, W. C. (1984). Discourse structures for text generation. Research Report ISI/RR-84-127, USC/Information Sciences Institute.

Mann, W. C. and Matthiessen, C. M. I. M. (1983). Nigel: A systemic grammar for text generation. Research Report ISI/RR-83-105, USC/Information Sciences Institute.

Mann, W. C. and Moore, J. A. (1981). Computer generation of multiparagraph english text. *Computational Linguistics*, 7(1):17–29.

McDonald, D. (1981). Natural language generation as a computational problem: An introduction. Technical Report 81–33, University of Massachusetts at Amherst.

McKeown, K. R. (1982). *Generating natural Language Text in Response to Questions About Database Structure.* PhD thesis, University of Pennsylvania.

McKeown, K. R., Wish, M., and Matthews, K. (1985). Tailoring explanations for the user. Technical Report cucs-172-85, Department of Computer Science, Columbia University.

Moore, R. (1981). Reasoning about knowledge and action. In Webber, N. J. N. B. L., editor, *Readings in Artificial Intelligence*, pages 473–477. Tioga, Palo Alto.

Patten, T. (1986). *Interpreting Systemic Grammar as a Computational Representation: A Problem Solving Approach to Text Generation.* PhD thesis, University of Edinburgh.

Perrault, C. R. (1989). An application of default logic to speech act theory. technical note 457, SRI International, Artificial Intelligence Center, Menlo Park.

Poole, D. (1988). A logical framework for default reasoning. *Artificial Intelligence*, 36:27–47.

Power, R. (1974). *A Computer Model of Conversation.* PhD thesis, University of Edinburgh.

Reichgelt, H. and Shadbolt, N. (1989). Planning as theory extension. In Cohn, A. G., editor, *Proceedings of AISB 89*, pages 191–199. Society for the Study of Artificial Intelligence and Simulation of Behaviour, Morgan Kaufman.

Reichman, R. (1985). *Getting Computers to Talk Like You and Me.* MIT Press, Cambridge Massachusetts.

Rosenschein, J. S. (1982). Synchronisation of multi-agent plans. In *Proceedings of the 2nd National Conference on Artificial Intelligence*, pages 115–119. American Association for Artificial Intelligence.

Sacerdoti, E. D. (1977). *A Structure for Plans and Behavior.* Artificial Intelligence Series. Elsevier Computer Science Library, New York, NY.

Searle, J. (1969). *Speech Acts: An Essay in the Philosophy of Language.* Cambridge University Press, Cambridge.

Shoham, Y. (1988). *Reasoning about change : time and causation from the standpoint of artificial intelligence.* series in artificial intelligence. MIT Press, Cambridge, Mass.

Sidner, C. L. (1983). Focusing in the comprehension of definite anaphora. In Brady, M., editor, *Computational Models of Discourse*, pages 267–330. MIT Press, Cambridge, Massachusetts.

Sperber, D. and Wilson, D. (1986). *Relevance.* Basil Blackwell, Oxford.

Steel, S. and Leung, E. (1989). Planning with ranges. In Cohn, A. G., editor, *Proceedings of AISB 89*, pages 213–223. Society for the Study of Artificial Intelligence and Simulation of Behaviour, Morgan Kaufman.

Strawson, P. (1964). Intention and convention in speech acts. *Philosophical Review*, 73:439–460.

Tate, A. (1976). Project planning using a hierarchic, non-linear planner. Technical Report 25, Dept. of AI, University of Edinburgh.

Warren, D. H. D. (1974). Warplan: A system for generating plans. Department of Computational Logic memo 76, Artificial Intelligence, University of Edinburgh.

Wilensky, R. (1983). *Planning and Understanding.* Addison-Wesley, Reading, Massachusetts.

Wilkins, D. E. (1988). *Practical planning : extending the classical AI planning paradigm.* series in representation and reasoning. Morgan Kaufmann, San Mateo, California.

Wilks, Y. and Ballim, A. (1987). Multiple agents and the heuristic ascription of belief. In *Proceedings of the eleventh International Joint Conference on Artificial Intelligence*, pages 118–124.

Wilks, Y. and Bien, J. (1983). Beliefs, points of view and multiple environments. *Cognitive Science*, 7:95–119.

Winograd, T. (1983). *Language as a Cognitive Process*, volume 1, Syntax. Addison-Wesley, Reading, Massachusetts.